BBN Systems and Technologies
A Division of Bolt Beranek and Newman Inc.

AD-A244 212

BBN Report No. 7352

SIMNET CVCC

# SIMNET SIMULATION OF RADIO COMMUNICATION: A TESTBED FOR INVESTIGATION OF C3I TECHNOLOGY

DTIC
S ELECTE D
JAN 08 1992
D

This document has been approved for public release and sale; its distribution is unlimited.

92-00301

92 1 6 085

Report No. 7352

# SIMNET Simulation of Radio Communication:
## A Testbed for Investigating Combat Vehicle C3I Technology

Arthur R. Pope, Ray Tomlinson, Jim Gonzalez, Dan Van Hook

**July 1990, July 1991**

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION UNLIMITED

**Prepared by:**

BBN Advanced Simulation,
a business unit of BBN Systems and Technologies
10 Moulton Street
Cambridge, MA 02138

**Prepared for:**

Defense Advanced Research Projects Agency
(DARPA)
Information and Science Technology Office
1400 Wilson Boulevard
Arlington, VA 22209-2308

U.S. Army Communications - Electronics Command
(CECOM)
Center for C3 Systems
Ft. Monmouth, NJ 07703

This research was performed by BBN Systems and Technologies under Contract Nos. MDA972-89-C-0060 and MDA972-90-C-0061 to the Defense Advanced Research Projects Agency (DARPA). The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policy, either expressed or implied, of DARPA, the U.S. Army, or the U.S. Government.

1991 Bolt Beranek and Newman Inc.

# Preface

The work described in this report was conducted on behalf of two programs: the SIMNET program, and the Combat Vehicle Command and Control (CVC2) program.

SIMNET is an advanced research project sponsored by the Defense Advanced Research Projects Agency (DARPA) in partnership with the United States Army. Currently in its seventh year, the goal of the program is to develop the technology to build a large–scale network of interactive combat simulators. This simulated battlefield provides, for the first time, an opportunity for fully–manned platoon–, company–, and battalion–level units to fight force–on–force engagements against an opposing unit of similar composition. Furthermore, it does so in the context of a joint, combined arms environment with the complete range of command and control and combat service support elements essential to actual military operations. All of the elements that can affect the outcome of a battle are represented in this engagement, with victory likely to go to that unit which is able to plan, orchestrate, and execute their combined–arms battle operations better than their opponent. Whatever the outcome, combat units will benefit from this opportunity to practice collective, combined arms, joint war fighting skills at a fraction of the cost of an equivalent exercise in the field.

While simulators to date have been shown to be effective for training specific military skills, their high costs have made it impossible to buy enough simulators to fully train the force. Further, because of the absence of a technology to link them together, they have not been a factor in collective, combined arms, joint training. SIMNET addresses both of these problems by aiming its research at three high–payoff areas:

- Better and cheaper collective training for combined arms, joint war fighting skills.
- A testbed for doctrine and tactics development and assessment in a full combined arms joint setting.
- A "simulate before you build" development model.

These payoffs are achievable because of recent breakthroughs in several core technologies that have been applied to the SIMNET program:

- High speed microprocessors.
- Parallel and distributed multiprocessing.
- Local area and long haul networking.

Dist | A-1

- Hybrid depth buffer graphics.
- Special effects technology.
- Unique fabrication techniques.

These technologies, applied in the context of "selective fidelity" and "rapid prototyping" design philosophies, have enabled SIMNET development to proceed at an unprecedented pace, resulting in the fielding of the first production units at Fort Knox, Kentucky, just three years into the development cycle.

In addition to the basic training applications, work is underway to apply SIMNET technology in the area of combat development to aid in the definition and acquisition of weapon systems. This is made possible because of the low cost of the simulators, the ease with which they can be modified, and the ability to network them to test the employment of a proposed weapon system in the tactical context in which it will be used, i.e., within the context of the combined arms setting.

The Combat Vehicle Command and Control (CVC2) is a program demonstrating the technology for an automated, integrated, and interoperable command, control, and communication system for ground combat vehicles. The $C^3$ system is to serve units at battalion level and below, and it is to link adjacent units and cross-attached units whether they are of the same or of allied nations.

The CVC2 program is developing and demonstrating data transmission protocols used for digitized voice and data communications. The challenges in this area include coping with the limited capacity and range of the communication medium (VHF FM radios), avoiding conflict between voice and data traffic sharing common networks, and providing for effective management of combat radio networks.

The work described here serves both programs. To the SIMNET program it represents the development and demonstration of a method for simulating radio phenomena in the context of a distributed simulation. To the CVC2 program it represents the development of a simulation testbed for investigating issues of communication among combat vehicles.

# Table of Contents

# 1. INTRODUCTION

This report describes an augmentation of the SIMNET distributed simulation system to include a simulation that represents important characteristics of radio communication. The radio simulation was developed for the U.S. Army's Communications – Electronics Command, to support the combat vehicle communication research they are conducting under the Combat Vehicle Command and Control (CVC2) program.

The SIMNET system creates a simulated world about a region of terrain typically thousands of square kilometers in area. Vehicles populate the simulated world where they operate under the control of soldiers and computers. To coordinate their actions soldiers and computers must be able to communicate by radio in the simulated world just as they do in the real world.

Most SIMNET vehicle simulators have been equipped with a simple radio system that makes use of commercial, citizens' band radios. Although relatively inexpensive to implement, this system has several important limitations as a simulation of Army VHF radios:

- [illegible] radio can communicate with any other, regardless of the distance or type of terrain between them;
- at most forty distinct frequencies are available for use;
- the radios may be used to communicate voice, but not data;
- there is no way of simulating detectors such as direction finding equipment or anti–radiation missiles; and
- the radio system does not permit a determination of which vehicle transmitted a particular signal.

In this report we describe a method of simulating radio communication which overcomes these limitations. The method serves equally well for both voice and data communication while providing the following capabilities:

- The locations of a transmitter and a receiver, and the shape of the terrain between them, affect radio reception. Our method simulates the attenuation of a radio signal due to distance and intervening terrain.

- Characteristics of the transmitter and of the receiver are taken into consideration. The transmitter's power, the receiver's sensitivity, and the antenna gain of each are all involved in determining received signal strength.
- Interference with radio signals, either intentional or otherwise, is an important factor on the battlefield. Our method permits simulation of radio interference, and allows jamming devices of various kinds to be simulated.
- Devices that detect a radio emission and locate its source—such as direction finding equipment or antiradiation missiles—are easily simulated with the method we propose.

We have tested and demonstrated this radio simulation method by implementing simulators for a family of VHF–FM combat network radios, called SINCGARS (Single Channel Ground and Airborne Radio System). These simulators have been installed at the Ft. Knox SIMNET-D site, where they provide radio communication among several vehicle simulators and a command post. The success of this demonstration has proven the efficacy of the radio simulation method described here.

The SINCGARS radio simulators at Ft. Knox are part of a simulation testbed that is used to investigate the application of information technology to the problems of combat vehicle command and control. The vehicle simulators of that testbed are equipped with a simulation of an experimental Intervehicular Information System (IVIS) that aids vehicle commanders in managing and communicating information. IVIS units in various vehicles exchange digital messages with each other by means of radio channels provided by simulated SINCGARS radios. As shown in Figure 1–1, each vehicle includes a simulated Radio Interface Unit (RIU) that mediates the use of radio channels by the IVIS unit in that vehicle. RIUs implement communication protocols in order to provide a communication service supporting IVIS. A simulation of RIUs is included in the SINCGARS radio simulators we have developed.

In summary, this report describes work in three areas:

- It describes a method of simulating radio communication. Chapter 2 of this report discusses the overall architecture of the radio simulation system: what major components exist, what those components do, and how they communicate. Chapter 3 provides a more detailed specification of the communication protocol used among radio simulation components.

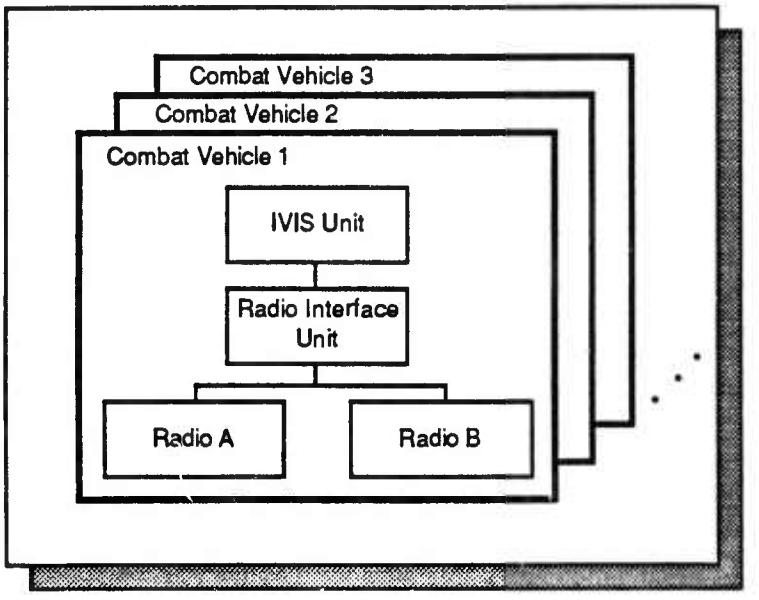


Figure 1–1. The simulation testbed includes simulations of the components depicted here. Each combat vehicle is equipped with an IVIS unit and one or two SINCGARS radios. A Radio Interface Unit provides an interface between the IVIS unit and the radios, while implementing communication protocols that serve to link IVIS units in adjacent vehicles. The SINCGARS radio simulator described in this report simulates both SINCGARS radios and their Radio Interface Units.

- It describes the implementation of a particular radio simulator built according to this method. The design of the SINCGARS radio simulator is described in chapter 4, while appendices A and B document details of how the simulator is configured and operated.
- It describes a simulation of a Radio Interface Unit, which has been incorporated into the SINCGARS radio simulator. The implementation of the RIU simulation is described in chapter 5.

The work described in this report was performed during the period January 1989 through May 1990, under the sponsorship of the U.S. Army's Communications – Electronics Command (CECOM) and the U.S. Defense Advanced Research Projects Agency (DARPA). Mr. Charles Graff of CECOM's Center for C3 Systems provided the guidance and technical information upon which this work was based. The SINCGARS radio simulator was developed at BBN by Jim Gonzalez, Arthur Pope, Ray Tomlinson, and Dan

Van Hook. From February 1990 through May 1990, the simulator endured daily use supporting experiments on the simulation testbed at Ft. Knox.

# 2. RADIO SIMULATION ARCHITECTURE

In this chapter we describe the architecture of the SIMNET radio simulation. By architecture we mean what major components are required, what tasks each of those components are responsible for, and how those components communicate.

The architecture described here is intended for the simulation of any VHF FM radio communication. It has been applied to the simulation of SINCGARS radios in particular, but it is believed to be of general usefulness. In this chapter, therefore, we describe the architecture in terms that are not specific to the SINCGARS implementation, postponing the details of that specific implementation to a subsequent chapter.

In developing this architecture, we have focused on the following goals and constraints:

- Any new simulation of voice radios must be compatible with the existing SIMNET radios so that radios of one type can communicate with those of the other (via an intermediary translation system if necessary).
- The radio simulation must be extensible in several ways. It must be reasonably easy to add additional radio simulators, to simulate new types of equipment, and to incorporate more sophisticated models of radio communication.
- The simulated radios must be able to communicate over the distributed simulation's long haul network. Moreover, the demands made on the long haul network for this communication must be minimized. In particular we wish to minimize the requirements for communication bandwidth, reliability, and delivery speed.
- The system must be structured so that any failure of equipment or software will have only a minimal effect on the overall operation of the system.
- Development, production, and operation costs must be minimized.

## 2.1 Modelling of signal propagation and reception

*Signal propagation*

Propagation of a radio signal from a transmitter to a receiver is affected by the characteristics of the transmitting and receiving antenna, the transmitted power and frequency, terrain shielding, diffraction, refraction, reflection, and a variety of

atmospheric phenomena. Because the SIMNET distributed simulation does not presently represent atmospheric phenomena (modelling always a cloudless, slightly–hazy day), the effects of atmospheric phenomena cannot yet be represented by a SIMNET radio simulation. The other factors are represented. In order to take them into account, the radio simulation must have access to the following information about the simulated world:

- for each transmitter, its location, power, antenna orientation, antenna characteristics, and frequency;
- for each receiver, its location, sensitivity, selectivity, antenna orientation, antenna characteristics, and tuning;
- for each transmitted message of speech or data, the identity of the transmitter and the contents of the message; and
- the shape of the terrain surface, and the nature of its covering.

In general, a propagation model can use this information to compute the strength of each radio signal received by a particular receiver. These signals will generally include one or more friendly transmitters and one or more enemy jammers. Of course, the same propagation models are appropriate to model the signal propagation to a receiver from a friendly transmitter or from an enemy jammer.

A number of signal propagation models have been developed for predicting the coverage of various classes of radio systems, operating in various environments and frequency bands. Some of these are surveyed in *Radio Wave Propagation: A Handbook of Practical Techniques for Computing Basic Transmission Loss and Field Strength* [1]. For example, a particularly simple model would be one that considers free–space attenuation only: the strength of a signal varies only as a function of the distance it travels from transmitter to receiver. A somewhat more elaborate model would examine the shape of the terrain surface between transmitter and receiver in order to estimate the contribution due to diffraction, absorption, scatter, etc. of the signal by intermediate obstacles.

Computational efficiency is an important consideration in the selection and implementation of a signal propagation model. To the extent that signal propagation is affected by the motion of a transmitter or receiver, it must be recomputed as either party moves. The model's sensitivity to location and orientation, and the degree of accuracy required in the application of the model, will determine how much motion may be tolerated before signal attenuation along a particular path must be recomputed.

*Signal reception*

In order to model the signals actually heard by the receiver operator, we must model the mutual interference effects of the multiple received signals in the nonlinear circuit ("detector") of the receiver. Again, one can choose among various approaches depending on the class of radio system to be simulated, and the degree of accuracy required.

For example, a simple model can be used to approximate the behavior of an FM detector. FM receiver circuits exhibit a phenomenon known as "capture" whereby they lock onto and track the strongest signal being received on a given frequency, and inhibit the effects of other signals that may be present. To model this, one could calculate which signal being received is strongest at any moment, and present the audio from this signal to the listening operator. This strongest signal may be either a friendly voice signal or hostile jamming, in which case an appropriately annoying noise will be presented to the listener. In the absence of any signal, the "squelch" circuit of the receiver inhibits all audio output. When a signal is being received in a "fringe" area where it is so weak that lock-on is marginal, it suffers rapid and erratic "drop out", which we will simulate by rapid interruptions of the audio output.

*Summary*

In summary, the radio simulation architecture should permit one to adopt any of various models to compute the effects of signal propagation and signal reception. The choice of models for a particular application will depend on the degree of accuracy required and the computational resources available. The architecture must ensure that the dynamically–changing information required by signal propagation and reception models is available for their use.

## 2.2 Distributed computation

*Network topology*

The goals of extensibility and reliability are best satisfied by a radio simulation system that is network–based and distributed. Individual simulators representing single radios or small groups of radios should be linked by a common, backbone network so that any simulator can communicate with any other.

This is precisely the architecture used by the SIMNET system for linking vehicle simulators. The architecture ensures that no single simulator, in failure, can cause the whole system to fail. Moreover, it allows the system to be extended readily by adding more simulators—and simulators of different types—to the backbone network, provided that the network's capacity is not exceeded.

A network is needed by the radio simulation to communicate the characteristics of transmitters and receivers, and the contents of radio signals. Of these, the radio signals can be expected to make up most of the requirement for network capacity. The network must have sufficient bandwidth to convey all of the radio signals being transmitted at any one time, and to do so with minimal delay. In normal operation, individual radios are operated in groups or networks such that only one of the radio transmitters participating in a radio network is transmitting at any one time. If we assume that the system will be used to simulate a series of radio networks in this way, then we can conclude that the required network capacity will scale linearly with the number of radio networks being simulated.

Depending on the network capacity needed, it may be possible to use a single physical network (e.g., Ethernet) to carry both vehicle simulation messages (such as vehicle appearance information) and radio simulation messages. Even when this cannot be done because the combined volume of vehicle and radio simulation messages exceeds the capacity of one physical network, it remains a simple matter to use two separate networks for the two types of messages. Figure 2–1 depicts the simpler case, in which a single network is used.

*Speech representation*

In this section we consider the issue of representing speech for communication among various components of the radio simulation system. Speech can be represented in various forms, both analog and digital. The most appropriate representation is chosen by considering the manner in which speech signals must be manipulated.

A problem that arises in simulating voice radios is that of selecting, for each receiver, just those voice messages originating from the transmitter currently being "heard" by that receiver. For example, when two transmitters are transmitting on the same frequency, any given receiver must be able to select between the two depending on which signal is being received more strongly at any moment. A simulated radio receiver must therefore

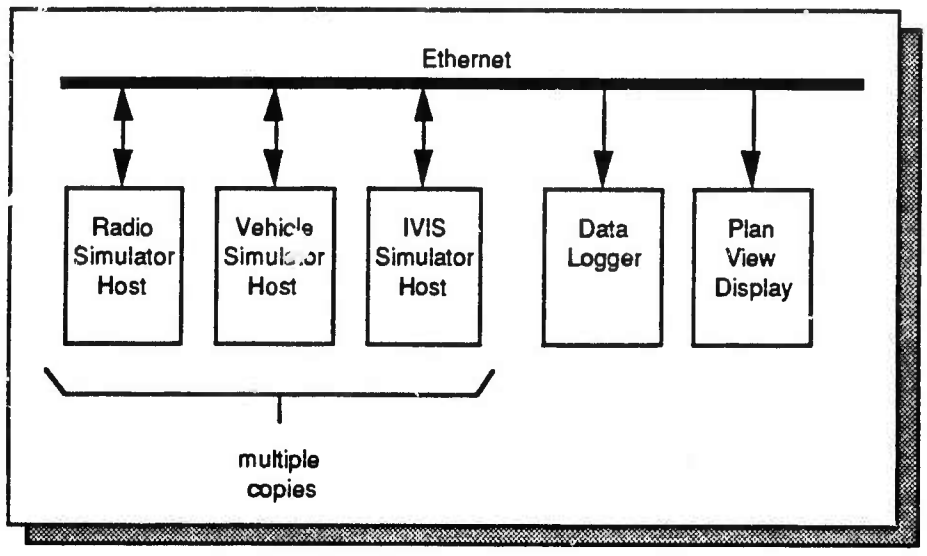


Figure 2–1. The radio simulation system can share the same backbone network used by other components of the SIMNET distributed simulation, provided that network has sufficient capacity.

be able to select, under software control, the appropriate signal from among different radio signals.

The manner in which the voice messages are represented greatly affects how easily this selection or switching of voice messages can be accomplished. If speech is communicated among radio simulators in analog form, then either the complexity of the interconnection network grows quadratically with the number of simulators, or a relatively complex radio simulator is required to select among multiplexed analog speech signals.

By digitizing the speech signals, one can use a simple network for interconnecting the radio simulators, and those radio simulators themselves can be relatively simple. Digitized speech can be communicated via Ethernet, which provides a natural way for each transmitter to broadcast messages containing digitized speech to all potential receivers. A receiver can then easily select just those messages that are from the transmitter it currently "hears".

Digitized speech signals afford other advantages for the radio simulation system as well. Conveying speech over a long–haul network can be done most efficiently and with least distortion if the speech is in digitized form. Once digitized, the speech can then be communicated over the same long–haul links as those used for vehicle simulation messages. Digitizing speech also simplifies the problem of recording it for later analysis.

Once digitized, speech can be recorded on the same media (and by the same computer system) as vehicle simulation messages, thus forming a more complete, consistent, and convenient record of a simulation exercise.

*Digital speech compression*

Once encoded in digital form, speech can be compressed to various degrees. Greater compression allows speech to be communicated with less network bandwidth, but it is achieved at the cost of increased computation that sometimes requires specialized signal or array–processing hardware.

Among the simplest techniques is *Mu–law* compression, which replaces each digital sample with an 8–bit code. This technique achieves toll quality (i.e., long–distance telephone quality) communication at 64,000 bits per second (64 kbps), with minimal computation and no specialized hardware.

A more elaborate technique, called the *APCHQ (Adaptive Pulse Code High Quality) algorithm*, can achieve toll quality communication at 16 kbps. The algorithm requires additional computation that is best done by specialized hardware. Such hardware is already in use within the SIMNET program to create long–haul links between citizens band radio simulation systems at various sites.

A technique called *CVSD (Continuously Variable Slope Delta)* is used by some military systems, including SINCGARS. It compresses speech to 16 kbps with somewhat less clarity than APCHQ, but the algorithm is less demanding computationally.

In choosing among these and other speech compression techniques, one must consider the cost of compression hardware, the resulting savings in network bandwidth, and the effect that compression may have on the quality of the speech signal.

*Computation of signal propagation and reception*

Computation of the model that determines the propagation and reception of a particular radio signal could be carried out in any of three places: at the transmitting radio's simulator, at the receiving radio's simulator, or in a separate system (called a *server*) that services many transmitters and receivers simultaneously.

A simple argument demonstrates that computing the model in the transmitting simulator is not the best approach. Since most radios are receiving at any one time, and relatively

few are transmitting, few radio simulators would be busy computing the model at any one time. However, (nearly) all radio simulators must be capable of computing the model, since any could potentially be used as a transmitter. This approach, therefore, requires that a large computing capacity be available, of which relatively little will be used at any one time.

The server approach has two drawbacks. Should the server fail for any reason, all radio communication may be disrupted. Secondly, the processing power of the server must grow quadratically with the number of radios it serves, and thus this approach does not permit easy expansion to large numbers of radios.

The best approach is to model, in each receiving radio simulator, the propagation and reception for just those signals being received by that radio. The radio simulator obtains from the network digitized voice messages broadcast by all other transmitters. Each message includes from ten to a hundred milliseconds of digitized speech, as well as information about the transmitter's location, orientation, frequency and power. Messages originating from transmitters too far away, and from transmitters tuned to the wrong frequency, can be discarded immediately by the receiver. The remaining messages represent signals from nearby transmitters that potentially can be heard, and the receiver models propagation and reception only for these relatively few signals.

To model propagation, each radio simulator must have information about the terrain that surrounds the receiver it simulates. If this terrain information is detailed, a correspondingly detailed model can be computed; if it is coarse, the model will be less precise. For simplicity, reliability, and efficiency, the terrain information is best stored in its entirety at each radio simulator. Where this is uneconomical because of the amount of overall terrain information, however, it is also possible to supply terrain information in pieces to the radio simulators, as needed, from "terrain servers" attached to the Ethernet. This terrain information can be far less detailed than that used by the computer image generators of manned simulators.

*Communicating radio location information*

So that it can model signal propagation a radio simulator must remain informed of the location of each receiver it simulates, and of each transmitter capable of transmitting to any of those receivers. These locations may be changing dynamically as radios are moved about in the simulated world. The distributed simulation system is already

conveying information about the dynamically–changing locations of vehicles; by associating each radio with a particular vehicle, we acquire a means of knowing the locations of radios as well.

Therefore each simulated radio is associated with a particular vehicle in the simulated world. By listening to messages describing the locations of vehicles, a radio simulator remains informed of the locations of any radios it is interested in. Messages that pertain to a particular radio—such as one containing a signal broadcast by that radio—identify the vehicle that radio is associated with so that the radio's location may be ascertained by any simulator receiving the message.

## 2.4 Frequency hopping

Simulation of a radio network employing frequency hopping deserves special consideration because each change of frequency may require some additional computation and communication by the radio simulators. There are various approaches to simulating frequency hopping. Each approach involves some tradeoff between the fidelity of the simulation, and the computation or communication necessary to support it. Two methods, which we shall call the "precise approach" and the "statistical approach", illustrate this tradeoff.

The precise approach seeks to model each change of frequency so that the simulator hops among frequencies just as the actual radio would. The pseudorandom sequence generator that determines the frequency at each new hop, and the algorithm that synchronizes a receiver with a hopping transmitter, are implemented in the simulator just as they are in the actual radio. This approach may produce the most accurate simulation of the system's performance in the face of interference and jamming, but it may also require the most computation by simulators. The algorithms for generating and synchronizing frequency hops, even though perhaps performed in hardware by the actual radios, would be performed in software by the simulators. This approach may also require the most communication among simulators. If, for example, each message contains speech transmitted on a single frequency, then a transmitter hopping 100 times per second (a relatively slow rate) would produce 100 messages per second. If, on the other hand, a message contains several "hops" worth of speech, then the processing each receiver must perform to examine the message is more complex. For these reasons, the precise approach to modelling frequency hopping is best suited to low hop rates, and to situations where the cost of greater fidelity is justified.

The statistical approach seeks to model the frequency hopping behavior at a higher level of abstraction while ensuring that the important characteristics of the behavior—such as susceptibility to jamming—are accurately represented. Because this approach does not simulate individual frequency hops, it cannot associate a precise frequency with each transmission. Instead, each transmission includes information that describes the overall hopping behavior, such as the set of frequencies being hopped among, the period of time between hops, and the starting point of the hop sequence. A given receiver, in determining whether a particular transmission is received, checks that its hop parameters match those of the transmission. If the same set of frequencies is being shared, even partially, by other transmitted signals, the receiver assesses a probability of jamming or interference, and corrupts or interleaves the incoming signals accordingly. This statistical approach to modelling frequency hopping may be the only approach viable at high hop rates.

## 2.5 Interworking with actual equipment

*Interconnecting simulated and actual radio networks*

A simulated combat radio network and a real one can be interconnected, as shown in Figure 2–2, to produce a single network that includes both real and simulated equipment. A *bridge*, consisting of a radio simulator and an actual radio, serves to relay transmissions between the two networks so that any radio may communicate with any other. A message transmitted by a radio simulator is received by the radio simulator half of the bridge, and relayed to the other half for retransmission into the atmosphere. Similarly, a message transmitted by an actual radio is received by the actual radio half of the bridge, and relayed to the radio simulator for retransmission onto the Ethernet.

Although this form of bridge allows interoperation between real and simulated radios, the radio propagation and reception model cannot operate correctly in the combined network. The system has no information about the locations of actual radios, nor does it know which actual radio may have transmitted a given message. Thus it cannot implement the propagation and reception model, which requires this information. As an alternative, the system might simply assume that each actual radio is able to communicate with all others, or that all actual radios are at some specified location.

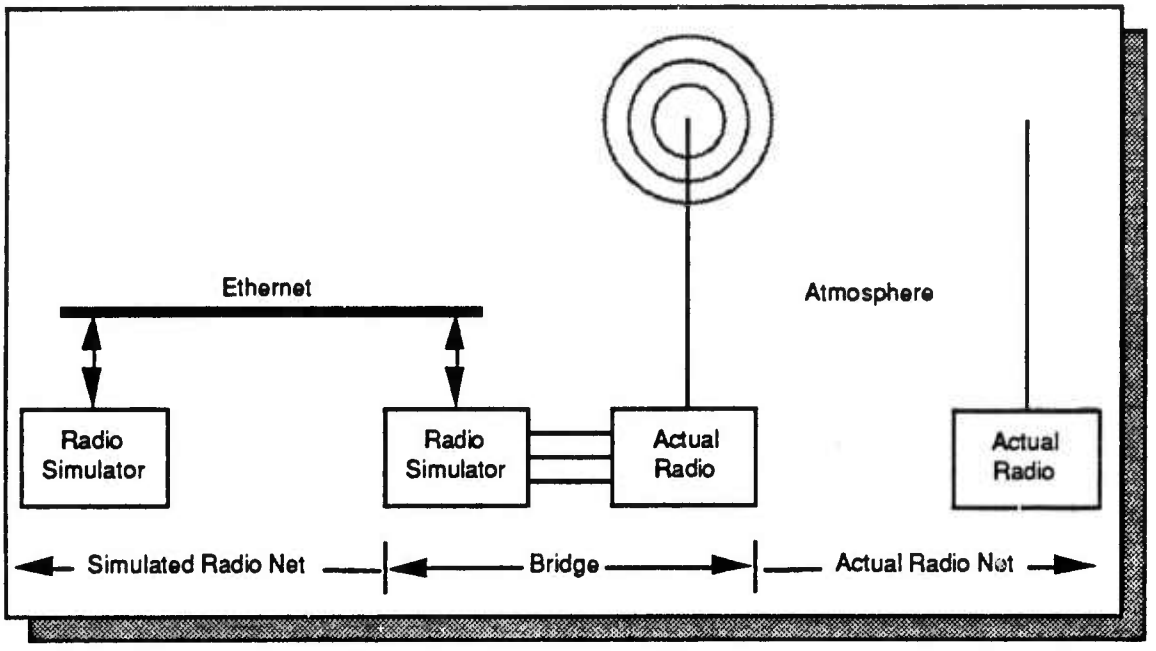


Figure 2–2. A bridge consisting of two components—a radio simulator and an actual radio—can join a simulated radio network and an actual radio network into a single net.

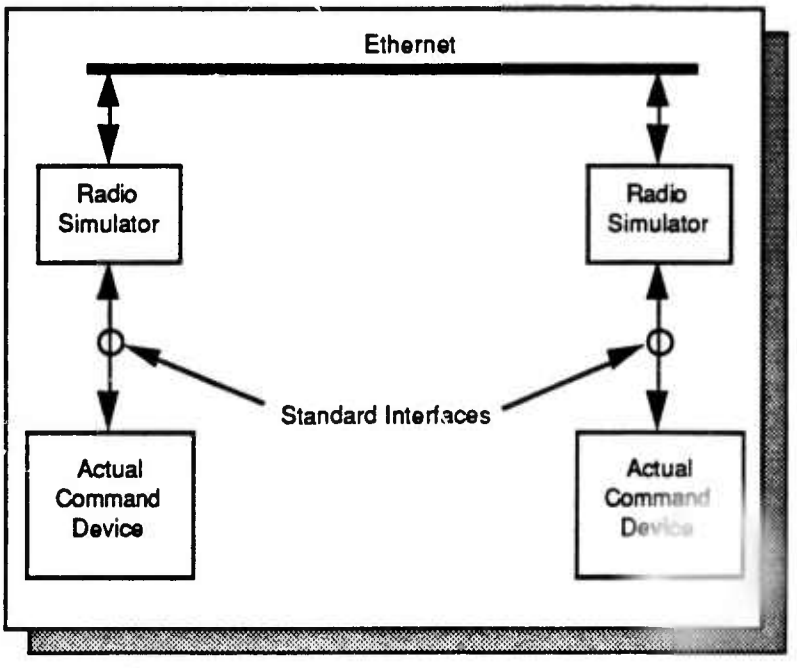


Figure 2–3. Actual command and control devices, such as portable computers, may communicate with each other using a simulated radio network.

*Incorporating actual command and control equipment*

Some command and control devices, such as portable computers, are designed to be connected directly to a tactical radio via a data or audio cable. A radio simulator can be designed to permit this same kind of connection so that actual devices may communicate using simulated radios. This possibility is depicted in Figure 2–3.

# 3. RADIO SIMULATION PROTOCOL

Radio simulator hosts communicate with each other—and report noteworthy events and statistics—by means of a *radio simulation protocol.* In terms of the ISO Basic Reference Model for Open Systems Interconnection [2], the radio simulation protocol is an Application Layer protocol.

The radio simulation protocol fits within a framework of distributed simulation protocols defined in the report, *The SIMNET Network and Protocols* [3]. In terms of the communication architecture defined in that report, the radio simulation protocol resides within the simulation sublayer, where it uses services provided by the underlying assocation sublayer.

In addition to sharing with other distributed simulation protocols a common communication architecture, the radio simulation protocol also shares various concepts. For example, the SIMNET protocols are based on concepts such as *exercise* (a simulation activity involving a group of simulators) and *vehicle identifier* (a unique identifier for a particular simulated vehicle). These concepts, which are employed by the radio simulation protocol, are defined in *The SIMNET Network and Protocols* [3]. That report also defines a notation called *Data Representation Notation*, which we use here to define the format of communication information.

Communication carried out according to the governing rules of the radio simulation protocol involves the exchange of units of data called *protocol data units* (PDUs). Each PDU is composed of individual data elements called *fields*. Among the first fields of every PDU is one that identifies the type of the PDU; that header is followed by additional fields whose format and meaning depend on the PDU's type.

In this section, we first define several data elements that are commonly used in various radio simulation PDUs. We then define the overall format of a radio simulation PDU, and describe how the underlying association sublayer service is used to communicate radio simulation PDUs. Finally, we discuss each communication activity mediated by the radio simulation protocol, and define the PDUs used to support these activities.

## 3.1 Concepts and common data elements

Each simulated radio (be it a receiver, a transmitter, or a receiver/transmitter combination) is associated with a particular vehicle present in the simulated world. For the purpose of computing the characteristics of a propagated signal, a radio's antenna is assumed to be at the same location as its vehicle. That location can be determined from information communicated among vehicle simulators as part of the simulation protocol described in *The SIMNET Network and Protocols* [3].

Each radio has a unique identifier that serves both to distinguish it from other radios, and to identify the vehicle with which it is associated. A radio identifier is of the following form:

```
type RadioID sequence {
    vehicle             VehicleID,              -- vehicle to which radio belongs
    radio               UnsignedInteger (8),    -- radio within vehicle
                        unused (8)
}
```

The `vehicle` component of this data element identifies the vehicle with which a radio is associated. It uniquely identifies a particular vehicle, as described in *The SIMNET Network and Protocols* [3]. So that each vehicle may have multiple radios, the `radio` component distinguishes among several radios belonging to one vehicle. The radios within a vehicle are numbered consecutively, from 0.

Several data elements are used to describe the nature of a transmitted signal. Their definitions, as presented in the following paragraphs, are sufficient to describe transmissions from SINCGARS radios; in future these definitions may be extended as needed to describe other types of transmissions.

The use that a signal makes of the frequency spectrum is described by a Radio Transmitter Mode data element:

```
type RadioTransmitterMode enum (8) {
    signal_SINCGARS_SC (1),                 -- SINCGARS single channel
    signal_SINCGARS_FH (2)                  -- SINCGARS frequency hopping
}
```

The manner in which a signal encodes information is described by a Radio Signal Type data element:

```
type RadioSignalType enum (8) {
    signalData (0),                    -- data bits
    signalAPCHQ16 (1),                 -- APCHQ 16kbps voice
    signalAPCHQ32 (2),                 -- APCHQ 32kbps voice
    signalCVSD16 (3),                  -- CVSD 16kbps voice
    signalERF (4)                      -- ECCM remote fill information
}
```

A signal's power, either as transmitted or as received, is described in units of dBm by a Radio Signal Power data element:

```
type RadioSignalPower float (32)
```

A signal transmitted on a single channel has a frequency described in units of Hertz by the following data element:

```
type RadioSignalFrequency UnsignedInteger (32)
```

The frequency hopping behavior of a radio is summarized by a FH Parameters data element:

```
constant numberFHLockouts 8

type FHParameters sequence {
    fh_hopset           UnsignedInteger (16),     -- hopset identification
    fh_lockout          array (numberFHLockouts) of UnsignedInteger (16),
                                                  -- lockouts being applied
    fh_transec          UnsignedInteger (16),     -- transmission security key
    fh_tod_offset       Time                      -- offset of TOD from world time
}
```

In this data element, hop sets, lockout sets, and transmission security (TRANSEC) variables are summarized by integer values. The meanings of these values (e.g., what specific frequencies are associated with a particular hopset value) are defined not by the

radio simulation protocol, but rather by the programming of the radio simulation hosts.[1] The `fh_tod_offset` component reports the discrepancy in seconds between a radio's own time–of–day clock and the precise time of day in the simulated world.

It may be desirable, for study purposes, to be able to identify the source of a radio transmission among crewmembers and computer systems within a vehicle. A Radio Speaker Identity data element is used for reporting this information:

```
type RadioSpeakerIdentity enum (8) {
    speakerUnknown (0),                    -- not known
    speakerCommander (1),                  -- tank commander
    speakerGunner (2),                     -- gunner
    speakerDriver (3),                     -- driver
    speakerLoader (4),                     -- loader
    speakerRIU (5),                        -- radio interface unit
    speakerERF (6)                         -- radio itself sending ERF data
}
```

## 3.2 Radio simulation protocol data units

The radio simulation protocol makes use of several kinds of protocol data units, which are summarized in Figure 3–1.

All radio simulation PDUs have a length that is an integral multiple of 64 bits, and all begin with a common 64–bit header. Included in this header is a code indicating the kind of PDU present:

```
type RadioPDUKind enum (8) {
    transmitterPDUKind (1),                -- Transmitter PDU
    receiverPDUKind (2),                   -- Receiver PDU
    signalPDUKind (3),                     -- Signal PDU
    intercomPDUKind (4),                   -- Intercom PDU
    alertOperatorPDUKind (5)               -- Alert Operator PDU
}
```

---

1 The present SINCGARS radio simulator relies on these frequency hopping parameters only to determine whether two radios are capable of communicating in the simulated world: if they have identical parameters then they are considered able to communicate.

| Protocol Data Unit | Purpose |
|---|---|
| Transmitter PDU | Describes a radio transmitter |
| Receiver PDU | Describes a radio receiver |
| Signal PDU | Communicates the content of a transmitted radio signal |
| Intercom PDU | Reports what is spoken over a vehicle's intercom system |
| Alert Operator PDU | Alerts a radio operator with an audible tone |

Figure 3–1. The radio simulation protocol employs the protocol data units summarized in this table.

Following the PDU header is a portion whose format depends on the kind of PDU. The overall content of a PDU is:

```
type RadioPDU sequence {
    version             RadioProtocolVersion,
    kind                RadioPDUKind,
    exercise            ExerciseID,
                        unused (40),
    variant             choice (kind) of {

      when (transmitterPDUKind)
        transmitter     TransmitterVariant,

      when (receiverPDUKind)
        receiver        ReceiverVariant,

      when (signalPDUKind)
        signal          SignalVariant,

      when (intercomPDUKind)
        intercom        IntercomVariant,

      when (alertOperatorPDUKind)
        alert           AlertOperatorVariant
    }
}
```

The `version` field specifies the version of the radio simulation protocol to which the PDU pertains. The use of this field allows new versions of the radio simulation protocol to be introduced without disruption to existing implementations. The radio simulation protocol described in this report has version number 1:

```
type RadioProtocolVersion enum (8) {
    radioProtocolVersionApr90 (1)
}
```

The `exerciseID` field identifies the exercise to which the PDU pertains, as described in *The SIMNET Network and Protocols* [3].

## 3.3 Use of association sublayer services

Radio simulation PDUs are conveyed among simulators using the services of the association sublayer defined in *The SIMNET Network and Protocols* [3]. In all cases, a single PDU is issued through a single invocation of the A–Datagram.req service primitive. This service primitive conveys a PDU from its issuing simulator to all other interested simulators.

To distinguish the radio simulation protocol from other protocols using the association sublayer, the radio simulation protocol is assigned a unique association sublayer user protocol number. This number is 134:

```
constant radioProtocolNumber 134
```

Communication involving the radio simulation protocol takes place within the context of a particular simulation exercise. Associated with an exercise is an exercise identifier, which distinguishes it from other, concurrent exercises. Protocol data units associated with a particular exercise are carried by the association service using a multicast group number that is identical to the exercise's identifier. This allows simulators to receive information only about the exercises of interest to them by subscribing only to selected multicast groups.

## 3.4 Communicating transmitter and receiver state information

*Transmitter PDU*

A radio simulation host reports the state of a transmitter it simulates by issuing a Transmitter PDU. This PDU must be issued whenever either (a) the state of the transmitter as described in the PDU changes, or (b) five seconds have elapsed since the last such PDU was issued. Transmitter PDUs are needed by hosts simulating radio receivers in order to determine the characteristics of received signals.

In addition to its PDU header, the Transmitter PDU includes the following fields:

```
type TransmitterVariant sequence {
    radio                RadioID,                  -- identity of radio transmitter
    transmitting         Boolean,                  -- whether transmitting signal
    changed              Boolean,                  -- whether PDU describes a change
                         unused(14),
    mode                 RadioTransmitterMode,     -- mode of transmitter operation
    speaker              RadioSpeakerIdentity,     -- identity of speaker, if any
    antennaHeight        Float (32),               -- height of antenna above terrain
    power                RadioSignalPower,         -- transmitter power (dBm)

    -- Depending on what mode the transmitter is in:
    u                    choice (mode) of {

      -- If SINCGARS single channel:
      when (signal_SINCGARS_SC) sincgars_SC sequence {
        frequency        RadioSignalFrequency      -- frequency in Hertz
      },

      -- If SINCGARS frequency hopping:
      when (signal_SINCGARS_FH) sincgars_FH sequence {
        hopinfo          FHParameters              -- frequency hopping parameters
      }
    }
}
```

The `radio` field identifies the radio transmitter described by this PDU, and the `transmitting` field indicates whether it is transmitting at the moment the PDU is issued. If the PDU is being sent because some attribute of the transmitter it describes has changed, then the `changed` field contains 1; otherwise it contains 0.

Even if the transmitting field indicates that the radio is not transmitting, the remaining fields of the PDU should contain values that describe how the transmitter would perform if it were transmitting. The mode field specifies whether the transmitter is in single channel or frequency hopping mode. Variants of the u field then describe, accordingly, the single frequency to which the transmitter is tuned, or the frequency hopping parameters characterizing the transmitter's use of the frequency spectrum.

The antennaHeight field specifies the height of the transmitter's antenna above the terrain, which must be known by other simulation hosts to compute the strength of a received signal. The power field specifies the power at which the transmitter's antenna is radiating energy while transmitting.

*Receiver PDU*

A radio simulation host reports the state of a receiver it simulates by issuing a Receiver PDU. This PDU must be issued whenever either (a) the state of the receiver as described in the PDU changes, or (b) five seconds have elapsed since the last such PDU was issued. Receiver PDUs are used to create a record of what a receiver "hears", for study purposes.

In addition to its PDU header, the Receiver PDU includes the following fields:

```
type ReceiverVariant sequence {
    radio               RadioID,            -- identity of radio receiver
    receiving           Boolean,            -- whether receiving signal
    changed             Boolean,            -- whether PDU describes a change
                        unused (30),

    -- If receiving a signal:
    power               RadioSignalPower,   -- received signal strength (dBm)
    transmitter         RadioID             -- identity of radio transmitter
}
```

The radio field identifies the radio receiver described by this PDU, and the receiving field indicates whether reception of a signal is occuring at the moment the PDU is issued. If the PDU is being sent because some attribute of the receiver it describes has changed, then the changed field contains 1; otherwise it contains 0.

The remaining fields of the PDU describe a particular signal being received by the radio. The power field specifies the strength of this signal at the receiver's antenna, and the

transmitter field identifies the radio transmitter issuing the signal. If reception is occurring, these fields describe the signal being received. Otherwise, if any signal whatsoever is reaching the receiver, these fields describe the strongest such signal (even though it is not being decoded successfully by the receiver). Finally, if no signal is reaching the receiver, these fields contain zero values.

## 3.5 Communicating radio signal information

*Signal PDU*

The content of a radio transmitter's signal over a brief period of time is described by a Signal PDU issued by that transmitter's simulator. While a tranmitter is broadcasting, its simulator repeatedly issues Signal PDUs to describe successive time segments of the transmitted signal. These PDUs are used by other radio simulators to decode a received signal.

In addition to its PDU header, the Signal PDU includes the following fields:

```
type SignalVariant sequence {
    radio               RadioID,                    -- identity of radio transmitter
    speaker             RadioSpeakerIdentity,       -- source of signal
    encoding            RadioSignalType,            -- type of signal
    synchronization     RadioSyncInfo,              -- message synchronization info
                        unused (8),
    time                UnsignedInteger (32),       -- time of start of signal segment
    duration            UnsignedInteger (16),       -- duration of signal segment (ms)
    bitcount            UnsignedInteger (16),       -- length of signal data, in bits
    dataLength          UnsignedInteger (16),       -- length of signal data, in octets
    data                array (dataLength) of Character (8)
                                                    -- signal data
}
```

The radio field identifies the radio transmitter broadcasting the signal described by this PDU, and the speaker field identifies the crewmember or system at that transmitter responsible for the transmission.

The encoding field specifies how the transmitted signal is encoded for inclusion within the Signal PDU; its values are defined in section 3.1, above.

The synchronization field specifies whether this segment of a transmitter's signal contains a preamble used to synchronize receivers with the transmitter. Its values are:

```
type RadioSyncInfo enum (8) {
    syncNormal (0),                    -- normal PDU
    syncPreamble1 (1),                 -- first preamble PDU
    syncPreamble2 (2),                 -- second preamble PDU
    syncPreamble3 (3),                 -- third preamble PDU
    syncPreamble4 (4),                 -- fourth preamble PDU
    syncEOM (127)                      -- end of message PDU
}
```

Each instance of a Signal PDU encodes a transmitted signal over a specific period of time. To aid a receiver's simulator in piecing together and decoding successive Signal PDUs, each Signal PDU identifies the period of time that the PDU covers. The PDU's `duration` field specifies the duration of its coverage, in milliseconds. The PDU's `time` field specifies the start time of the signal segment, also in milliseconds. The `time` field is not based on the value of a global clock known to all radio simulators; instead, it is derived from the transmitting simulator's local clock. Consequently the `time` field is consistent only among Signal PDUs pertaining to the same radio transmitter. Nevertheless, its presence allows a receiving simulator to detect missing Signal PDUs, and to pace its processing of the Signal PDUs it obtains.

The `data` field contains the segment of encoded signal rounded up in size so that the size of the entire PDU is a multiple of 64 bits. The length of the `data` field in octets is given by the `dataLength` field. The length of the meaningful portion of the `data` field in bits is given by the `bitcount` field.

## 3.6 Other functions

Two PDUs described in this section serve special purposes outside the scope of radio simulation. However these PDUs have been included in the radio simulation protocol because they are implemented specifically by radio simulators, and because the manner in which they are encoded has much in common with other radio simulation PDUs.

*Intercom PDU*

A radio simulator may be privy to what is spoken over an intercom internal to a vehicle as well as that which is spoken for radio transmission.[2] For study purposes it may be desirable to record this intercom speech. The Intercom PDU is used to report intercom speech via the simulation network so that is may be recorded by a system such as the SIMNET Data Logger. In addition to its PDU header, the Signal PDU includes the following fields:

```
type IntercomVariant sequence {
    vehicleID           VehicleID,              -- vehicle containing speaker
    speaker             RadioSpeakerIdentity,   -- source of signal
    encoding            RadioSignalType,        -- type of signal
    time                UnsignedInteger (32),   -- time of start of signal segment
    duration            UnsignedInteger (16),   -- duration of signal segment (ms)
    bitcount            UnsignedInteger (16),   -- length of signal data, in bits
    dataLength          UnsignedInteger (16),   -- length of signal data, in octets
    data                array (dataLength) of Character (8)
                                                -- signal data
}
```

The `vehicleID` field identifies the vehicle whose intercom speech is being reported, and the `speaker` field identifies the individual speaking on that intercom, if known. The remaining fields of the Intercom PDU serve functions similar to indentically named fields of the Signal PDU.

*Alert Operator PDU*

A radio simulator may be capable of emitting a warning or alerting tone through a simulated radio receiver's speaker or headset. If so, the Alert Operator PDU can be used to elicit that tone. In addition to its PDU header, the Signal PDU includes the following fields:

---

2 Such is the case, for example, with the SINCGARS radio simulators, which obtain speech signals from each crewmember's microphone regardless of whether those crewmembers are transmitting on a radio.

```
type AlertOperatorVariant sequence {
    radioID                 RadioID                 -- identity of radio to generate tone
}
```

The PDU's `radioID` field identifies the particular radio that is to emit the alerting tone.

# 4. SINCGARS RADIO SIMULATION

The two previous chapters have discussed a general architecture for radio simulation, and a communication protocol used to support that architecture. Now, in this chapter, we describe a particular implementation of a radio simulation that has been constructed according to that architecture. This implementation, which has served to verify and demonstrate the architecture, is of the SINCGARS family of VHF FM radios.

This chapter is organized as follows. Section 4.1 describes the SINCGARS family of actual radio equipment. Section 4.2 describes the features of our simulation of that equipment. The hardware we use for this simulation is described in section 4.3, and much of the software providing this simulation is described in section 4.4. A portion of the software that determines the strength of a propagated radio signal is described separately, in section 4.5. The software that simulates Radio Interface Units is described in the next chapter rather than this one although it is executed on the same hardware as the software described here.

## 4.1 Overview of SINCGARS

SINCGARS (Single Channel Ground and Airborne Radio System) is a new family of VHF–FM combat network radios designed to provide the primary means of command and control for combat, combat support, and combat service support units in the Army. SINCGARS radios improve on the previous generation of tactical radios by providing more channels, better communication security, better ECCM capability, and greater reliability. The SINCGARS family includes a series of modular components—such as a receiver transmitter (RT), a power amplifier, remote controls, and antennas—that can be combined in different configurations to produce a variety of "manpack" and vehicle–borne radios. These components and their application are described in documents [4] through [7] listed in the references section of this report.

The SINCGARS system is widely applicable, and its adoption will ensure widespread interoperability among tactical radios. Because SINCGARS will someday be the primary means of communication within the Army brigade, this system is an appropriate choice among tactical radios for SIMNET simulation.

Two RT models (designated RT–1439 and RT–1523) currently form the basis for both manpack and vehicular SINCGARS radios. These RTs are, of course, compatible with the

communication standards that must be met by all SINCGARS components to ensure interoperability. The characteristics of either RT model is summarized as follows:

- The RT provides 2320 channels at 25 kHz increments from 30 MHz to 88 MHz. To further reduce the effects of jamming, it can be tuned 5 or 10 kHz away from a channel's center frequency.
- It can operate in either a single channel mode or a frequency hopping mode. When frequency hopping, the RT changes frequency about 100 times per second, selecting frequencies according to a pseudorandom sequence determined by a key and other parameters.
- The operator can establish six channel presets. The RT can scan among these six channels, as well as a manually entered frequency and a "cue" channel, for any transmissions.
- Voice is transmitted as either an analog signal (for interoperability with previous–generation radios), or as a digital signal. The Continuously Variable Slope Delta (CVSD) algorithm is used to digitize speech at 16 kbps.
- The RT transmits data at bit rates of 75 bps to 16 kbps, and it can mix both voice and data on a single channel while distinguishing between the two. Data is communicated to and from the RT over a MIL–STD–188–100/114 interface (a specification comparable to EIA RS–232).
- The RF power output can be selected from among 32 μW, 160 mW, and 4W. The typical range of the RT equipped with a transmitter power amplifier, as used in a vehicle, is 35 km for voice and somewhat less for 16 kbps data.

Although the two RT models are quite similar, the RT–1523 model is the newer of the two models and it incorporates some improvements and additional features. Principle among these is an integral communication security (COMSEC) device. Whereas an external COMSEC device must be used with the RT–1439 to encrypt and decrypt voice and data, such a device is included in the RT–1523. The operator's interface of the RT–1523 also differs from that of the RT–1439 in various minor ways. The operator's interfaces for the two RTs are described in the manuals [6] and [7].

A SINCGARS RT may be used in conjunction with a transmitter power amplifier, designated model AM–7238. When this power amplifier is present a selection switch on

the RT's front panel determines whether the amplifier is active; when active, the amplifier delivers about 50 W of power to a transmitter's antenna.

SINCGARS system components may be configured in a great variety of ways to meet a range of applications. Of these configurations, two are of particular interest for the work described in this report:

- A vehicle may be equipped with a "long–range" configuration designated VRC–90. One VRC–90 includes a single RT–1523 receiver/transmitter and a single AM–7238 power amplifier.
- Alternatively, a vehicle may be equipped with a "long–range/short–range" configuration designated VRC–91. One VRC–91 includes two RT–1523 receiver/transmitters and a single AM–7238 power amplifier. The power amplifier is connected to just one of the two RTs.

Both configurations include a mounting bracket, which encapsulates the devices comprising the configuration and supplies power to them. An antenna is present for each RT.

## 4.2 SINCGARS features simulated

The SINCGARS radio simulator simulates an RT–1523 in either of two physical environments:

- As a "standalone" VRC–90 radio set, which may be placed on a tabletop in a simulated command post. In this configuration, speech is input through a handheld microphone, and output through a speaker or headset.
- As a VRC–91 radio set installed in an M1 tank simulator, with operator controls situated at the loader's station. Speech is input and output through the vehicle's internal intercom system. The M1 simulator is described in *SIMNET M1 Abrams Main Battle Tank Simulation: Software Description and Documentation (Revision 1)* [8].

In either case, each simulated RT–1523 is represented by a front panel that closely resembles that of the actual RT–1523. The controls and indicators needed to operate functions implemented by the radio simulator are present, operating as they would for the actual radio. Other features of the front panel not relevant to the simulator, such as

connectors and a display dimmer control, are simply represented by silkscreened images of those features.

These controls are present on the simulated RT–1523 front panel: RF power, function, channel, mode, COMSEC, and volume. They differ from those of the actual radio in that knobs are not pulled out to change control settings. Also, the whisper mode of the volume control (achieved by pulling the volume knob out) is not supported. A 16–key keypad and 8–character LED display are also present on the simulated front panel.

The simulation allows certain characteristics of each simulated RT to be determined by entries in a configuration parameter file. For example, an entry determines whether a particular RT has a associated AM–7238 transmitter power amplifier. Appendix A defines the contents of the configuration parameter file.

The following is a survey of the RT–1523 operating procedures and features supported by the radio simulation.

*Transmitter power selection.* The RF power switch is fully functional. Its power level setting is used to determine the simulated transmission power. This, in conjunction with the radio propagation model, is used to determine the strength of received signals, which in turn determines which signal is heard and the accuracy of the received data.

Whether an AM–7238 power amplifier is present may be specified for each radio by means of an entry in the simulator configuration file. Radios without a power amplifier transmit with the same signal strength in both the HI and PA positions.

*Channel selection.* The channel selector is fully functional and determines which frequency or hopset is used for transmission and reception. A receiver can only receive a signal which is transmitted on the frequency (in single channel mode) or hopset (in frequency hopping mode) to which it is tuned. Other signals are treated as noise and contribute to an interference level that may cause data corruption to occur.

*Function selection.* The function selector is functional in the following positions: STBY, OFF, LD, SQ-ON, SQ-OFF, and Z-FH. The TST position (initiating a self–test by the RT) and REM position (allowing control of the RT from a remote control–monitor) are not functional. The LD function is supported only for ECCM remote fill (ERF) operations. Local fill is not simulated.

*Mode selection.* The mode selector is fully functional. The radio operates in single channel mode when the switch is in the SC position and operates in frequency hopping mode when the switch is in the FH or FH–M positions. Selecting the FH-M position enables the use of NCS master functions, such as transmitting ERF information.

*Signal strength.* A bargraph signal strength indicator displays transmitted power while the radio is transmitting, and received power at other times.

*Single channel frequency entry.* The operating procedures for loading, storing, clearing, and displaying a single channel frequency are supported by the simulation. Also supported is the procedure for offsetting a single channel frequency by ±5 or ±10 kHz.

*Single channel mode.* The simulation allows communication in single channel mode. In this mode, two radios will communicate only if they are operating on the same single channel frequency and offset.

*Frequency hopping mode.* The simulation allows communication in frequency hopping (FH) mode. For each radio, a collection of eight lockout sets is maintained; for each channel, a hopset is maintained. Lockout sets and hopsets are identified by integers that bear no meaning apart from serving as unique identifiers. Switching a radio to FH mode on a specific channel selects a hopset identifier and a collection of lockout set identifiers for transmitting and receiving. Each radio also maintains a per channel time-of-day, and a TRANSEC (transmission security) variable. A FH transmission can only be received by a radio if the hopset and lockout set match, if the time-of-day is within 4 seconds, and if the TRANSEC variable matches.

FH mode exhibits a capture phenomenon wherein once the reception of a message commences, it is relatively insensitive to subsequent transmissions even if they utilize the same hopset and lockout sets; the subsequent transmission is treated as an interfering signal with a low probability of interfering with the received signal.

*Frequency hop data entry.* FH data (such as hopsets and lockout sets) can be supplied either from a simulator configuration file, or by an ECCM remote fill operation. A radio in FH–M mode (NCS master) can transmit hopset and lockout set information to other radios (typically using the MAN channel). This information can be stored by receiving radios and used for subsequent FH operation. The present radio simulation does not support late net entry so time of day must be established manually.

There is no provision for simulating the loading of FH data from an ECCM fill device.

*COMSEC (communication security).* The COMSEC selector is present on the front panel but not functional. The simulation behaves as though this switch were set to its PT position regardless of its actual setting. Operator functions that relate to the RT–1523 models internal COMSEC features, such as the "*" function and display of the COMSEC key, are not implemented by the simulation.

*Cue frequency.* Monitoring of the CUE channel is simulated. Each idle radio periodically "listens" on the frequency specified for its CUE channel (every 1.9 seconds whenever the radio is not actively transmitting or receiving). A transmission of sufficient signal strength causes "CUE" to appear on the front panel display.

*Battery condition.* Operating procedures for setting battery condition, and RT features for reporting battery condition, are not simulated.

*TOD (time–of–day) clock.* The TOD clock is simulated. Operating procedures for setting and displaying the TOD clock are supported. The simulation requires TOD clocks to be synchronized for successful FH communication. The present simulation does not simulate TOD clock drift.

*Data rate.* Operating procedures for setting and displaying the data rate are not supported. A simulated data rate is established automatically by the RIU simulation.

*Channel scanning.* Channel scanning, in which the receiver repeatedly scans up to eight SC frequencies, is not simulated.

*Retransmit operation.* Retransmit operation, in which a pair of RTs are cabled together to serve as a communication relay, is not simulated.

*RT self–test.* The self–test function of the RT is not simulated.

## 4.3 Simulator hardware

The overall radio simulation comprises one or more radio simulators that are connected by a simulation Ethernet. Each radio simulator implements one or more individual radios, each configured either for installation in an M1 vehicle simulator or for standalone operation.

In general the radio simulator comprises the following hardware subsystems:

- a host computer that models signal propagation and reception and emulates the front panel operations of the radios;
- a speech I/O subsystem that collects speech input and converts it to compressed digital form, and performs the inverse operation for speech output;
- a simulation Ethernet interface allowing the host computer to communicate with other radio simulators; and
- a set of facsimile radio front panels resembling those of the actual radio, together with interface electronics linking controls and indicators on those front panels to the host computer.

*Host computer*

A Concurrent Computer Corp. MC6600 system was selected to be the radio simulator host computer for the following reasons:

- The system runs a version of UNIX that is enhanced to support real–time applications. The enhancements include a provision for fixing the priority of a process so that it may intentionally monopolize a processor and thereby provide prompt response to real–time events.
- The system has the hardware features needed to process the signal propagation model, such as floating point hardware. It also has industry standard VMEbus and Multibus interfaces allowing it to incorporate other radio simulator subsystems.
- The processing power of the system can be increased, if needed, by the addition of more processors and more memory.
- The same type of system also serves as host to other SIMNET systems, including combat vehicle simulators and command–and–control system simulators. The choice of a common system means that certain software can be reused, and fewer varieties of spare parts must be stocked at SIMNET sites.

The MC6600 system is based on a Motorola MC68030 microprocessor, clocked at 25 MHz. The chassis of this system contains a Multibus backplane with space for several I/O interface boards. When configured to be a radio simulator host, the system also includes the following components:

- 8 MB of main memory;
- 280 MB of disk storage;
- a VMEbus backplane with space for eight VMEbus cards; and
- two serial I/O multiplexor boards, each supporting eight serial I/O ports that may be configured for various data rates and signalling protocols.

*Speech I/O subsystem*

The speech I/O subsystem uses a VMEbus interface board to convert analog speech signals to and from compressed, digitized form. The board is termed *SIMVAD*, for *SIMNET Voice Analog/Digital*. Each SIMVAD board supports two full–duplex speech I/O channels; it has a 40 MHz TI TMS320C25 signal processor, A/D and D/A convertors, memory, and control logic dedicated to each channel. The analog I/O connections for each channel are made through a DB–9 style connector mounted on the board's front panel. Compression and decompression of speech signals is performed by the signal processors, under control of firmware residing in PROM on the board. To input and output a digitized speech signal, the host computer reads and writes registers on the board that are accessible via its VMEbus interface.

The speech I/O subsystem also includes analog signal conditioning circuitry. One form of conditioning circuitry is used for a radio in standalone configuration. This circuitry, which is located with the radio front panel, interfaces to a microphone jack and a speaker.

A second, and more elaborate, form of conditioning circuitry is used for a pair of radios installed in an M1 vehicle simulator. In this case, speech signals are input from taps on crewstation microphones so that both radio and intercom transmissions are detected. Radio speech signals are output to the vehicle's intercom system, which distributes them to crewstation headsets. The signal conditioning circuitry performing these functions for one M1 vehicle simulator (outfitted with two radios) is packaged as a board called the *Vehicle Intercom Adapter (VIA)*. A VIA board is installed near the intercom system with which it interfaces.

*Simulation Ethernet interface*

Installed in the host computer's Multibus chassis is a controller board allowing communication via the simulation Ethernet. The controller board, which is

Communication Machinery Corporation's model ENP–30, provides hardware support for the Ethernet physical and data link layer protocols. The board has a Motorola MC68020 microprocessor and 128K bytes of RAM that can be programmed to support protocols at higher layers. When used in a radio simulator host, this board is programmed to receive relevant simulation Ethernet packets, and to copy these packets to buffers in host computer memory for processing by the radio simulation program.

*Front panel subsystem*

Each radio is operated using a facsimile of its front panel. For a standalone radio, the front panel forms one face of an enclosure that may reside on a tabletop. For a vehicle–installed radio, the front panel is one of two that form a panel installed at the loader's station of the M1 vehicle simulator.

The front panel contains several rotary selector switches whose positions are sensed by a nearby digital interface board called an *Interactive Device Controller* (IDC)[3]. Under the control of a Z80 microprocessor, the IDC board repeatedly detects the position of each switch and reports any changes to the simulator's host computer. The IDC board communicates with the host via a serial data cable connected to one of the ports on the host's serial I/O multiplexor. For a standalone radio, the IDC board also senses and reports the position of the microphone push–to–talk (PTT) switch. In a vehicle installation, one of the two IDC boards present for the two radios in the vehicle senses and reports the positions of each crewmember's intercom selector switch and PTT switch.

The keypad and character display on the front panel are support by a second digital interface board, called a *Front Panel Adapter* (FPA). The FPA, which is located directly behind the front panel, contains a serial interface for transmitting keystrokes to the host computer, and for receiving from the host computer characters to be displayed. A serial data cable connects the FPA to one of the ports on the host's serial I/O multiplexor.

---

3 The IDC board was originally developed for use in SIMNET vehicle simulators, to link switches, control handles, and indicators to simulator host computers.

*Summary*

The outlying components of a simulated radio—its front panel, microphone, and speaker—are linked to the radio's simulator host computer by several cables. Two serial data cables communicate a radio's switch positions, keypresses, and display characters. From two (for a standalone radio) to six (for a pair of vehicle–installed radios) audio cables communicate speech signals. Each cable may be up to 150 feet in length.

The host computer described here allows support by one simulator of up to eight radios in any mix of standalone and vehicle–installed configurations. However the frequency with which signal propagation strength is recomputed will decrease as the number of radios supported is increased. Consequently, in scenarios where radios are moving rapidly in the simulated world, it may be desirable to limit to fewer than eight the number of radios supported by an individual simulator. The tradeoff between the number of radios supported and the accuracy of signal propagation modelling, under various simulated conditions, has not yet been accurately determined.

## 4.4 Simulator software

*Process overview*

The radio simulation software consists primarily of a single application process that executes on each radio simulator's host computer. This process simultanously supports all of the radios implemented by that simulator. The process is responsible for:

- maintaining the state of each radio it simulates;
- maintaining knowledge of the vehicles within which those radios reside;
- maintaining knowledge of what transmitters exist, where those transmitters are located in the simulated world, and what each transmitter's characteristics are;
- maintaining current estimates of signal attenuation over each path, from every transmitter to each receiver it simulates;
- responding to inputs from radio front panel switches and keypads while updating the displays and internal radio state information accordingly;
- accepting digitized speech signals from the voice I/O subsystem, and issuing Signal and Intercom PDUs containing the signals;

- accepting Signal PDUs from the network (and from the host's own transmitters), simulating signal detection and capture by each receiver, and supplying digitized speech signals to the voice I/O subsystem for output; and
- supporting an interface to the RIU simulation which allows each simulated RIU to sense channel activity, transmit data, and receive data.

Because these activities are all carried out by a single process, any overhead due to context switching among processes is avoided. The single process organizes its own agenda in order to perform promptly any operations that are time–critical, such as moving speech signal information between the speech I/O subsystem and the network interface. The process performs other operations, such as updating estimates of signal attenuation, in the background with whatever time is not taken up by the time–critical processing.

Each SIMVAD speech I/O channel processes one *frame* of incoming speech and one frame of outgoing speech every 26.25 milliseconds. This establishes the pace for all of the host process' time–critical operations: every 26.25 milliseconds it must service each speech I/O channel, perhaps accepting an incoming frame and perhaps providing an outgoing one. In the context of the radio simulator, this 26.25 millisecond cycle is referred to as a *tick*.

Each new tick begins when the host process receives an interrupt from a particular SIMVAD board. One SIMVAD board—that with the numerically lowest channel number—is programmed to generate this interrupt whenever it has prepared a new frame for input. Since all speech I/O channels process incoming and outgoing frames at the same rate a single interrupt suffices to initiate the servicing of all channels.

Upon receiving an interrupt marking the start of a new tick, the host process performs the following activities:

- It performs any periodic processing required for the simulation of RIUs.
- It reads frames from each speech I/O channel, and issues those frames as Signal PDUs and/or Intercom PDUs.
- It checks for and processes any PDUs received from the network. For certain PDUs it updates its knowledge of simulated vehicles, transmitters, or the state of its own radios. The contents of Signal PDUs bearing radio signal information may be passed to one or more speech I/O channels.

- Occasionally it determines whether any input has been received from a radio front panel. In response to such input, it will determine that radio's new state, and perhaps output a new string of characters for display on the radio's front panel.

This processing is initiated by an interrupt, rather than by periodic polling, to ensure that it will occur promptly at the required time.

When not performing this interrupt–based processing, the host process is cycling through the following series of operations:

- It checks for any terminal keyboard input. The terminal used to start the host process may be used to issue commands to it during its execution. These commands, which are meant primarily for debugging, are described in Appendix B.
- It recomputes signal attenuation for the paths between selected pairs of transmitters and receivers. This computation is described in the following section.
- It periodically sends PDUs that report the state of its radio transmitters and receivers.
- It checks that various hardware interfaces are responsive, and resets those that are not.

*Signal reception*

At any particular moment, each receiver is considered to be either synchronized with a particular transmitter and receiving from it, or not synchronized with any transmitter.

The received power from the transmitter to which the receiver is synchronized (or the strongest transmitter if it is not synchronized) is considered. If it does not exceed the sum of noise and interference power from other sources, then the receiver will become de–synchronized from the transmitter (or will not become synchronized). This means that if a number of transmitters, each of which would not individually interfere with a reception, are transmitting simultaneously, they can jointly interfere with reception.

In the present simulation, noise power is a constant -116 dBm (which is a parameter readily modified through a configuration file). This represents the inherent noise of the receiver. It does not include any ambient noise received on the antenna or otherwise.

Interference is modelled by simply adding the power from all sources other than the desired transmitter. The contribution from an individual interference source depends on its frequency and mode. There are four cases: SC to SC, SC to FH, FH to SC, and FH to FH. For SC to SC interference the model is that the interference power is attenuated by 35 dB for the first 25 kHz separation and by an additional 5 dB for each additional 25 kHz separation.

For all other cases, the interference power is assumed to be attenuated by 31 dB. This is a crude approximation whose justification is that with a typical hopset consisting of 1200 frequencies, the probability that one transmission will coincide in frequency with another is 1 in 1200 and therefore the interference power will be attenuated by 1/1200.[4]

If a receiver is not synchronized with a transmitter and that receiver's squelch has been turned off (FCTN switch in the SQ OFF position), then frames of a previously recorded random noise sound source are played out through the receiver's audio output. Similarly, whenever a data signal (i.e., an ERF or RIU transmission) is being received a pure tone is played out through the receiver's audio output.

## 4.5 Radio signal propagation model

The signal propagation model used by the radio simulation is the Longley–Rice model, which is fully described in *A Guide to the Use of the ITS Irregular Terrain Model in the Area Prediction Mode* [9]. The radio simulation incorporates a C language implementation of this model that was derived from FORTRAN code provided by CECOM. The model is valid for the frequency range and separation distances involved, and is well-suited to the rural environment normally represented in SIMNET simulations.

It should be noted that this model is not suitable for urban environments, where steep take-off angles (the angle between the antenna base and the horizon) are frequently encountered as vehicles approach tall buildings. Such steep take-off angles cause the model to break down due to trigonometric errors. Other models have been considered for use in urban environments, but no such environments currently exist in SIMNET to warrant the implementation of these models.

---

4 $10 \cdot \log\frac{1}{1200} = -31$ dB.

The model requires information about terrain elevation along the straight line between a transmitter and a receiver in order to develop a terrain "roughness" measure. To compute this measure, the simulation uses an array of terrain elevation values residing in main memory. This two–dimensional array has an elevation value for each 50 meter by 50 meter region of terrain. Each elevation value is obtained by averaging the ground elevation every 10 meters within its region, and adding the height of the tallest "electromagnetically significant" structure found within the region. The electromagnetically significant structures are taken to include any building that is more than 10 meters across at its widest point—a standard that omits telephone poles, small trailers, and individual trees.

An array of elevation values is prepared from a larger and more complete collection of terrain information called a *terrain database*. The extraction process, which takes several hours, need only be performed once for each terrain database. A separate program extracts the array of elevation values and writes it to a file, in preparation for its use by the radio simulation.

The array of elevation values for a 50 km by 75 km area of terrain, such as that described by the Ft. Knox terrain database, occupies 3 megabytes of main memory. In contrast the entire terrain database, which includes more detailed elevation data as well as information about covering features, occupies 35 megabytes. Thus the discrete, 50-meter sampling is necessary so that the entire array of elevation values will reside in the radio simulator host's main memory, permitting rapid access. We have found no gross inaccuracies in our implementation of the propagation model due to this discrete, 50–meter sampling of terrain elevation information.

Computation of the signal attenuation between a particular transmitter and receiver can take a significant amount of time on the radio simulation host processor. For example, in a worst–case scenario where the two are located at opposite corners of a 50 km by 75 km terrain area (90 kilometers, or 1803 fifty-meter regions, apart), the computation takes 119 milliseconds. Most of this time is used for the calculation of the terrain roughness measure described above. To avoid needless calculation, attenuation between a particular pair of transmitter and receiver is repeated only when either has moved by a significant distance. As a background processing task, the radio simulation repeatedly checks transmitter/receiver pairs and recomputes the signal attenuation for those that require it.

The radio simulation includes a feature for bypassing the signal propagation model for specific radios. It allows any radio to be declared a "perfect radio" via a configuration file entry. A perfect radio is one that may transmit to any receiver and receive from any transmitter, regardless of the intervening distance or terrain.

# 5. RADIO INTERFACE UNIT SIMULATION

In a vehicle such as the M1 tank, data communication using SINCGARS radios will be mediated by an intelligent controller called a Radio Interface Unit (RIU). Each vehicle will be equipped with a single RIU that is connected to one or more radios, as shown schematically in Figure 1-1. The RIUs will format and queue data for transmission on radio channels, perform error checking and correction of received data, and implement schemes for addressing, routing, and network management. The RIUs of the various vehicles participating in a single combat radio network will behave as packet-switching nodes to provide, collectively, a data network for communication among vehicles. In addition, some RIUs may serve as bridges or gateways, linking two or more individual combat radio networks to permit communication among vehicles on distinct networks.

The SIMNET radio simulation includes a simulation of a prototypical RIU. We have deliberately made this RIU simple so that it will represent one end of a spectrum of possible RIU capabilities. Briefly, this RIU mimics the error correction and addressing conventions of the Maneuver Control System (MCS) Segment 11 protocol [10]; it does not, for example, support multi-hop routing, internetwork transfers, or automated network management. Section 5.1 of this chapter describes the simulated RIU in more detail. Future plans call for enhancements of the simulated RIU in order to investigate the effectiveness of additional capabilities.

This RIU simulation has been designed specifically to provide for transfer of messages among SIMNET IVIS simulators. Consequently, the services provided by the RIU are invoked through an IVIS-to-RIU interface. The nature of this interface, which is described in section 5.2, provides some insight into the operation of the RIU simulation.

Another view of the RIU simulation is obtained by looking at the communication protocol used among RIUs. This protocol uses simulated SINCGARS radio channels as its medium. It is described in section 5.3.

The final section of this chapter contains some discussion of the RIU simulation's internal operation.

## 5.1 Features

The RIU simulation provides the interface to the SINCGARS radio for data transmission and reception by an IVIS unit. It simulates the various protocols used to transfer data from one IVIS unit to another. The present RIU implementation supports five modes in which this data may be delivered:

| | |
|---|---|
| reliable point–to–point | single recipient with retransmission. |
| reliable multicast | multiple recipient with retranmission. |
| unreliable point–to–point | single recipient without retransmission. |
| unreliable multicast | multiple recipient without retransmission. |
| unreliable broadcast | all recipients without retransmission. |

Each IVIS unit is identified by a unique address. Data conveyed by RIUs may be directed to a single address (point–to–point), to a set of addresses (multicast), or to any who may happen to receive the data. If the recipients' addresses are known then the data may, optionally, be conveyed reliably by the RIUs through the use of acknowledgements and retransmissions. Otherwise, only one delivery attempt is made and no acknowledgements are sought.

The data conveyed consists of blocks of characters (i.e., octets) in a private message format of the IVIS simulation. These messages are not formatted according to any established standard such as MCS Segment 11; rather, they are formatted according to the specific needs of the IVIS simulation. However, for the purposes of simulating radio channel performance, the RIU can simulate the effects of message lengths which are arbitrarily different from their actual length. For example, if an IVIS simulation message consists of 112 characters, but it would be carried by a 288 character MCS message, then the simulation can pad the message to occupy a duration of 288 character times on the radio channel.

This padding accommodates variations due to both message format and error control encoding. The present version pads all messages with 5 more octets for every 8 octets of data. This represents the expansion due to an assumed error detection and correction code.

The radio simulator performs message corruption by computing a bit error rate based on received signal strength and interference strength. The bit error rate determines how many data characters are corrupted. The algorithms for computing bit error rate from

signal and noise strengths in the present implementation are qualitatively correct, but they do not attempt to simulate the actual performance of the SINCGARS radio. The functions implementing these algorithms are place-holders for a more accurate model when available.

Error correction is performed in the same manner as provided by MCS Segment 11. Unacknowledged messages are retranmitted and all retransmitted copies of a message are merged until no erroneous characters remain. Retransmission is periodic up to a maximum number of attempts as in MCS Segment 11.

## 5.2 IVIS–to–RIU Interface

A single host computer simulates both an RIU and the collection of radios with which it is associated. Section 4.3 of this report describes the host computer hardware. Up to four RIUs may be simulated by a single host computer.

An IVIS unit is simulated by a separate host computer that communicates with its corresponding RIU by means of the simulation Ethernet. The IVIS–to–RIU interface, therefore, is in the form of a protocol that operates over the simulation Ethernet between a radio simulation host and an IVIS simulation host. We call this protocol the *IVIS protocol*. Like the radio simulation protocol described in chapter 2, the IVIS protocol is an Application Layer protocol conveyed by an underlying association sublayer service. IVIS protocol data units (PDUs) are defined in a manner analogous to radio simulation PDUs, using Data Representation Notation (DRN).

*Basic data elements*

We begin our description of the IVIS–to–RIU protocol by defining some of the basic data elements communicated. Each IVIS unit is identified by a unique address that is represented by an IVIS System Identifier data element:

```
type IVIS_SystemIdentifier VehicleID
```

(The Vehicle ID data element is defined in *The SIMNET Network and Protocols* [3].) As the notation implies, an IVIS unit's address is identical to the 48–bit vehicle identifier of the vehicle in which that IVIS unit resides. Since this identifier is unique for each vehicle it is also unique for each IVIS unit.

The various combat radio networks used by RIUs are also named by identifiers. These identifiers are of the following form:

```
type IVIS_NetworkIdentifier sequence {
    battalion           UnsignedInteger (8),     -- battalion operating network
    network             IVIS_NetworkType         -- type of network within battalion
}
```

The `battalion` component is simply an integer identifying the battalion within which the network is operating; values of these integers are selected during the setup of a simulation exercise. The `network` component specifies a particular network within that battalion:

```
type IVIS_NetworkType enum (8) {
    bnCommandNetwork,                    -- battalion command network
    aCoNetwork,                          -- A company's network
    bCoNetwork,                          -- B company's network
    cCoNetwork,                          -- C company's network
    dCoNetwork,                          -- D company's network
    a1PltNetwork,                        -- A company's 1st platoon network
    a2PltNetwork,                        -- A company's 2nd platoon network
    a3PltNetwork,                        -- A company's 3rd platoon network
    b1PltNetwork,                        -- B company's 1st platoon network
    b2PltNetwork,                        -- B company's 2nd platoon network
    b3PltNetwork,                        -- B company's 3rd platoon network
    c1PltNetwork,                        -- C company's 1st platoon network
    c2PltNetwork,                        -- C company's 2nd platoon network
    c3PltNetwork,                        -- C company's 3rd platoon network
    d1PltNetwork,                        -- D company's 1st platoon network
    d2PltNetwork,                        -- D company's 2nd platoon network
    d3PltNetwork                         -- D company's 3rd platoon network
}
```

Currently, network identifiers are used only to tag an IVIS message with the identity of the network on which it is being transmitted to simplify later analysis.

The unit of data conveyed among IVIS units by a collection of RIUs is called an *IVIS message*. Since the RIUs neither examine nor modify the contents of the IVIS messages they convey, the format of IVIS message content is not relevant to this discussion of the RIUs. For our present purposes the IVIS message can be considered to be simply an opaque data element of variable length:

```
type IVIS_Message sequence (
    -- various internal fields
)
```

The current simulation permits an IVIS message to be as large as 232 octets.

*IVIS protocol data units*

Three kinds of IVIS PDUs are involved in the IVIS–to–RIU interface. They are summarized in Figure 5–1 and identified by the following codes:

```
type IVIS_PDUKind enum (8) (
    ivisTransmitRequestPDUKind (4),
    ivisTransmitResponsePDUKind (5),
    ivisReceivePDUKind (6)
)
```

| Protocol Data Unit | Purpose |
|---|---|
| Transmit Request PDU | IVIS simulator requests communication of a message by an RIU |
| Transmit Response PDU | RIU reports to an IVIS simulator the results of trying to communicate a message |
| Receive PDU | RIU conveys a received message to an IVIS simulator |

Figure 5–1. The IVIS–to–RIU interface is a protocol involving the protocol data units summarized in this table.

Each IVIS PDU contains a common header followed by a variable part whose format depends on the kind of PDU:

```
type IVIS_PDU sequence {
    version              IVISProtocolVersion,
    kind                 IVIS_PDUKind,
    exercise             ExerciseID,
                         unused (40),
    variant              choice (kind) of {

      when (ivisTransmitRequestPDUKind)
        transmitRequest  IVISTransmitRequestVariant,

      when (ivisTransmitResponsePDUKind)
        transmitResponse IVISTransmitResponseVariant,

      when (ivisReceivePDUKind)
        receive          IVISReceiveVariant
      }
}
```

The header includes a protocol version number, which, for the version described in this report, is 1:

```
type IVISProtocolVersion enum {
    ivisProtocolVersionApr90 (1)
}
```

The header also includes the identifier of the simulation exercise in which the IVIS simulator and radio simulator are both participating.

*Transmit Request PDU*

An IVIS simulator sends a Transmit Request PDU to its corresponding radio simulator when it wishes to convey an IVIS message to one or more other IVIS simulators. The PDU is communicated from IVIS simulator to radio simulator using the association sublayer's transaction service. The PDU is issued as a transaction request; the corresponding transaction response carries no data.

In addition to its PDU header, the Transmit Request PDU includes the following fields:

```
constant maxRecipients 2
```

```
type IVISTransmitRequestVariant sequence {
    radio               RadioID,                    -- identity of radio transmitter
    network             IVIS_NetworkIdentifier,     -- network carrying message
    serialNumber        UnsignedInteger (16),       -- request serial number
                        unused (14),
    broadcast           Boolean,                    -- whether meant for all receivers
    acknowledge         Boolean,                    -- whether to be acked by receivers
    priority            UnsignedInteger (8),        -- priority of communication
    numberRecipients    UnsignedInteger (8),        -- number of specified receivers
    recipient           array (maxRecipients) of IVIS_SystemIdentifier,
                                                    -- list of recipients
    message             IVIS_Message                -- message
}
```

The `radio` field identifies a particular radio through which the message is to be transmitted by the RIU. Of course this radio must be one residing in the same vehicle as the IVIS unit and the RIU. The `network` field identifies the network on which the message is to be transmitted. It must be the network to which the specified radio is currently tuned.

Using the `serialNumber` field, the IVIS simulator may identify its particular request so that it can later match it with a subsequent report on that request (which comes in the form of a Transmit Response PDU from the radio simulator, described below).

The `broadcast` and `acknowledge` fields together specify the mode of communication requested. If the `broadcast` field contains 1, the message will be transmitted to all IVIS units on the specified network. Otherwise, it will be transmitted only to the set of IVIS units specified by the `numberRecipients` and `recipient` fields. If the `acknowledge` field contains 1, the message is to be conveyed reliably to its recipients using acknowledgement and retransmission among RIUs. Otherwise, no such measures are to be taken by RIUs to improve communication reliability. The `broadcast` and `acknowledge` fields may not both contain 1.

The `priority` field specifies the relative priority of this message for communication. Larger values correspond to higher-priority messages. Finally, the `message` field contains the content of the message to be conveyed to other IVIS simulators.

*Transmit Response PDU*

Using a Transmit Response PDU, a radio simulator tells an IVIS simulator the result of an RIU's efforts to convey an IVIS message. Each Transmit Request PDU from an IVIS

simulator to a radio simulator engenders a corresponding Transmit Response PDU in the reverse direction. The PDU is communicated from radio simulator to IVIS simulator using the association sublayer's transaction service. The PDU is issued as a transaction request; the corresponding transaction response carries no data.

In addition to its PDU header, the Transmit Response PDU includes the following fields:

```
type IVISTransmitResponseVariant sequence {
    radio               RadioID,                    -- identity of radio transmitter
    network             IVIS_NetworkIdentifier,     -- network carrying message
    serialNumber        UnsignedInteger (16),       -- request serial number
    transmitted         Boolean,                    -- whether message transmitted
                        unused (7),
    numberRecipients    UnsignedInteger (8),        -- number of failing receivers
    recipient           array (maxRecipients) of IVIS_SystemIdentifier
                                                    -- list of failing recipients
}
```

The `radio`, `network`, and `serialNumber` fields of the Transmit Response PDU contain the same values as the fields of the corresponding Transmit Request PDU.

The `transmitted` field indicates whether the RIU was successful at least in transmitting the IVIS message onto a radio channel. If reliable communication to a specified set of recipients was requested, the `numberRecipients` and `recipient` fields list those recipients that did not acknowledge successful receipt of the entire IVIS message.

*Receive PDU*

A Receive PDU conveys a received IVIS message from a radio simulator to an IVIS simulator. The PDU is communicated using the association sublayer's transaction service. It is issued as a transaction request; the corresponding transaction response carries no data.

In addition to its PDU header, the Receive PDU includes the following fields:

```
type IVISReceiveVariant sequence {
    radio               RadioID,                    -- identity of radio receiver
    network             IVIS_NetworkIdentifier,     -- network carrying message
    message             IVIS_Message                -- message
}
```

The radio field identifies the receiving vehicle's radio through which the message was received. The network field identifies the radio network on which the message was communicated. The message field contains the content of the message.

## 5.3 RIU–to–RIU communication

A protocol is used among RIUs in order to forward data [illegible] return acknowledgements. The protocol involves the use of datagrams that are communicated as data over simulated SINCGARS radio channels. Two kinds of datagram are used:

```
type RIU_DatagramKind enum (8) {
    riu_DG_Message,                          -- a transmit request
    riu_DG_LinkAck,                          -- a link acknowledgement
}

type RIU_Datagram sequence {
    sender              IVIS_SystemIdentifier,    -- who sent the message
    network             IVIS_NetworkIdentifier,   -- network carrying message
    serialNumber        UnsignedInteger(16),      -- request serial number
    kind                RIU_DatagramKind,         -- which variant
                        unused (8),
    variant             choice (type) of {

      when (riu_DG_Message)
        message         IVISTransmitRequestVariant,

      when (riu_DG_LinkAck)
        recipient       IVIS_SystemIdentifier,
    }
}
```

An RIU datagram of type riu_DG_Message is issued by an RIU to transmit an IVIS message at the request of an IVIS simulator. An RIU datagram of type riu_DG_LinkAck is used to acknowledge receipt of such a message by another RIU. In both cases, the sender field identifies the system sending the IVIS message (which, in the case of a link acknowledgement, is <u>not</u> the system sending the datagram).

For compactness, the RIU Datagram does not repeat the addressing information contained in the IVIS Transmit Request Variant. An exception to this is the sender field, which is required for both the riu_DG_Message variant and the riu_DG_LinkAck variant.

## 5.4 RIU simulation software

On behalf of each RIU, the simulation maintains a block of storage for each RIU from which it has received a message or to which a non-broadcast message has been directed. The connection structure provides for message sequencing and for simulated error correction through merging.

The RIU simulation uses or simulates the use of error recovery techniques specified by MCS Segment 11. This includes a retransmission strategy of a limited number of retransmission attempts spaced uniformly in time (i.e., no backoff strategy is employed) and message correction by the simulated use of error correcting codes and by merging the correct portions of several retransmissions. The current simulation performs retransmissions up to four times (for a total of five transmissions), at intervals of five seconds.

It should be noted that a simulated RIU can only have one message undergoing transmission at a time. A Transmit Response PDU for a message undergoing transmission will not be returned to an IVIS simulator until all link acknowledgements have been received or all retries performed. If another Transmit Request PDU is received from an IVIS simulator prior to the Transmit Response PDU for the former message, the transmission of that message is aborted, a Transmit Response returned to IVIS simulator indicating the aborted outcome (e.g. not transmitted), and transmission of the new message is begun.

The RIU inserts a random delay from zero to 700 milliseconds prior to the start of transmission of a link acknowledgement or the start of transmission of a message which was preempted or deferred (by a voice transmission or active reception). This delay is necessary to avoid the situation where all the radios with pending data transmissions start to transmit simultaneously. Since every radio will delay by a different amount, one will start first and be heard by the others which will thus not start transmitting and will not interfere with the transmission.

Another timer is used to provide a simple method for synchronizing serial numbers when a new radio rejoins the network. Ordinarily, the RIU only accepts messages whose serial numbers are greater than previously received messages. A message with a serial number which is less than or equal to the serial number of the last accepted message is discarded as

being a duplicate (usually due to retransmission caused by a lost acknowledgement). However, if a RIU (or IVIS unit) is restarted, the previous serial number is lost. To handle this situation, the RIU considers any sequence number to be acceptable after two minutes have elapsed without receiving an acceptable message. This time interval is sufficient to allow old duplicates to die out.

The radio simulator performs message corruption by computing a bit error rate based on received signal strength and interference strength. The strength of a received signal and of the sources interfering with that signal are computed as described in section 4.4. The ratio of signal power to noise and interference power is converted into a bit error rate (BER) using a simple table lookup. The present simulation uses a table populated with representative values; more accurate values derived from detailed models or empirical measurements can be incorporated into the table in future.

A signal/noise of 0 dB or less produces a BER of 0.5 (completely random). Above 0 dB, the BER is as described by the table in Figure 5–2.

For a given BER, the expected number of corrupted octets is computed and the corruption of that number of data octets is simulated. The present simulation implements a 13/8 coding rate and assumes a per data octet Hamming code. A data octet has to have two or more bits in error to be considered incorrect. The effect of undetected errors is ignored. A perfect Time Dispersion Coding method is assumed such that bit errors are totally independent.

| Signal/Noise (dB) | Bit Error Rate |
|---|---|
| 0 | 0.50 |
| 1 | 0.45 |
| 2 | 0.40 |
| 3 | 0.30 |
| 4 | 0.20 |
| 5 | 0.10 |
| 6 | 0.05 |
| 7 | 0.02 |
| 8 | 0.004 |
| 9 | 0.001 |
| 10 | 0.0003 |
| 11 | 0.0001 |
| 12 | 0.00003 |
| 13 | 0.00001 |
| ... | ... |

Figure 5–2. This table shows the bit error rates assumed for various possible values of signal–to–noise ratio.

# 6. REFERENCES

[1] U.S. Department of Defense. *Radio Wave Propagation: A Handbook of Practical Techniques for Computing Basic Transmission Loss and Field Strength.* DoD publication ECAC–HDBK–82–049.

[2] International Standards Organization. *Information processing systems — Open Systems Interconnection — Basic Reference Model.* ISO 7498-1984.

[3] Arthur Pope. *The SIMNET Network and Protocols.* BBN Report Number 7102. BBN Laboratories, Inc. Cambridge, Mass., July 1989.

[4] *SINCGARS Staff Planning Guide.* PM SINCGARS, CECOM, Ft. Monmouth, NJ, November 1987.

[5] *SINCGARS System Engineering Document — Ground Radio System (Third Draft).* CECOM, Ft. Monmouth, NJ, February 1989.

[6] U.S. Army. *Technical Manual TM 11–5820–890–10–1, Operator's Manual, Radio Sets AN/PRC–119, Advance Copy.* Headquarters, Dept. of the Army, July 1986. (This operator's manual pertains to the RT–1439 SINCGARS receiver/transmitter.)

[7] *Operator's Narrative for the SINCGARS ICOM Radio RT–1523(C)/U.* 14 September 1988. (This operator's manual pertains to the RT–1523 SINCGARs receiver/transmitter. It's authorship and publisher are unknown.)

[8] James Chung, Alan Dickens, Brian O'Toole, and Carol Chiang. *SIMNET M1 Abrams Main Battle Tank Simulation: Software Description and Documentation (Revision 1).* BBN Report Number 6323. BBN Systems and Technologies Corp. Cambridge, Mass., August 1988.

[9] National Technical Information Agency. *A Guide to the Use of the ITS Irregular Terrain Model in the Area Prediction Mode.* NTIA Report 82–100.

[10] Ford Aerospace and Communications Corp. *Maneuver Control System (MCS) Top Level Interface Control Document (ICD).* CECOM, Ft. Monmouth, NJ, 18 April 1986.

# APPENDIX A: SIMULATION CONFIGURATION OPTIONS

At startup the SINCGARS radio simulation program obtains configuration information from two sources: a text file of configuration parameters, and a series of "switches" or options included with the command used to invoke the program. This appendix defines the information obtained from each source.

## A.1 Configuration parameter file

The configuration parameter file describes each radio to be implemented by a simulator, including the radio's capabilities and the hardware resources associated with it. The file also provides information about the propagation environment, and it directs the simulation to information about the terrain. The file is usually present in the simulator's UNIX filesystem as */simnet/data/sincgars/pars*.

Each line, or *entry*, in the configuration file consists of a keyword followed by one or more parameters. Several different kinds of entries are recognized; each is described below. There are some restrictions on the order in which entries of various kinds may appear in the file. The order in which they are described below satisfies all of these restrictions.

Certain entities described by the configuration file, such as simulated radios and simulated RIUs, are identified by small integer indices. These indices are used to identify the following entities:

*radio* — Each radio simulated by the radio simulation host is identified by a unique integer in the range [0,7].

*vehicle* — Each vehicle simulator having radios simulated by the radio simulation host is identified by a unique integer in the range [0,3]

*voice channel* — Each voice digitization channel is identified by a unique integer in the range [0,15]. Each SIMVAD voice digitization board has two channels; these are identified by two consecutive indices. The first (even numbered) index refers to the channel associated with the board's upper DB–9 analog I/O connector; the second (odd numbered) index refers to the channel associated with the lower DB–9 connector.

| | |
|---|---|
| *RIU* | Each RIU simulated by the radio simulation host is identified by a unique integer in the range [0,7] |
| *ivis* | Each RIU simulated by the radio simulation host is associated with one or more IVIS units, simulated by a separate host. Each such IVIS unit is identified by a unique integer in the range [0,7]. |

The following paragraphs describe entries present in the configuration parameter file.

*Exercise*

The *exercise* entry identifies the simulation exercise in which the simulator is to participate. There is only one such entry in a simulator's configuration file.

```
Format:  exercise    exercise-id
Example: exercise    3
```

*Terrain*

The *terrain* entry specifies a file containing an array of terrain elevation values. There is only one such entry in a simulator's configuration file.

```
Format:  terrain     filename
Example: terrain     /simnet/data/sincgars/Knox.0305.rdb
```

*Antenna*

The *antenna* entry specifies the antenna transmit and receive gains in dB, and the antenna height in meters above the terrain. There is only one such entry in a simulator's parameter file; it applies to all the radios implemented by that simulator.

```
Format:  antenna     transmit-gain receive-gain height
Example: antenna     0.0  0.0  3
```

*Environment*

The *environment* entry provides various parameters characterizing the signal propagation environment. These are a climate number (as defined for the Longley–Rice model), refractivity measure, soil conductivity measure, and permittivity measure. There is only one such entry in a simulator's configuration file; it applies to all radios implemented by that simulator.

```
Format:  environment climate refractivity conductivity permittivity
Example: environment 4  301  0.005  15.0
```

*Sensitivity*

The *sensitivity* entry specifies a receiver's sensitivity, in dBm. This is the minimum received power needed for a receiver to "hear" a signal. There is only one such entry in a simulator's configuration file; it applies to all radios implemented by that simulator.

```
Format:  sensitivity sensitivity
Example: sensitivity -116
```

*Voicechannel*

A *voicechannel* entry describes a single speech I/O channel. The *VME-address* parameter specifies the VME bus address of the hardware I/O registers for the channel. The *encoding-scheme* parameter specifies the method to be used by the channel for encoding and decoding speech: `CVSD` or `APCHQ_16Kbps`.

```
Format:  voicechannel      voice-channel VME-address encoding-scheme
Example: voicechannel      0  e000  APCHQ_16Kbps
```

*Radio*

A *radio* entry defines a radio to be simulated. The *position* parameter specifies the position of the radio within a vehicle simulator, with 0 representing the radio A position, and 1 representing the radio B position. (For a standalone radio, the *position* parameter should be 0.) The *IDC-port* parameter specifies the serial port (0 through 7) to which the radio's IDC board is connected. The *display-port* parameter specifies the UNIX "tty" port (as `/dev/tty8` through `/dev/tty15`) to which the radio's Front Panel Adapter is connected. The *voice-channel* parameter specifies the voice channel for radio speech output and, in the case of a standalone radio, also for speech input. If the `amp` keyword is present, the radio is simulated as having a transmitter power amplifier. If the `perfect` keyword is present, the radio enjoys perfect (i.e., unaffected by distance or terrain) communication with other radios.

```
Format:  radio      radio position IDC-port display-port
                    voice-channel [ amp | perfect ]
Example: radio      0  0  0  /dev/tty8  0  amp
```

The configuration file contains one *radio* entry for each radio implemented by a simulator.

*Vehicle*

The *vehicle* entry describes a single M1 vehicle simulator containing radios implemented by the radio simulator. The *IDC-port* parameter specifies the serial port (0 through 7) to which an IDC board is connected. That IDC board reports the positions of the vehicle's intercom selector switches and push-to-talk switches. The *number-radios* parameter specifies how many simulated radios the vehicle has; it is followed by that many radio numbers identifying those radios. If the `log` keyword is present, the simulator will report all speech conveyed over the vehicle's intercom system by issuing Intercom PDUs.

```
Format:  vehicle     vehicle IDC-port number-radios { radio } [ log ]
Example: vehicle     0  0  2  0  1  log
```

The configuration file contains one *vehicle* entry for each M1 vehicle simulator in which the radio simulator implements a radio.

*Station*

The *station* entry describes a single crewstation within an M1 vehicle simulator. The *vehicle* parameter identifies that vehicle, and the *station* parameter names the station within it: `commander`, `loader`, `gunner`, or `driver`. The *voice-channel* parameter specifies the voice channel receiving speech from the microphone at that crewstation.

```
Format:  station     vehicle station voice-channel
Example: station     0  commander  0
```

The configuration file contains one station entry for each crewstation from which the radio simulator receives speech input.

*Attach*

The *attach* entry associates a simulated radio with a vehicle present in the simulated world. There are four ways of defining this association, and four corresponding forms of *attach* entry.

An *attach* entry of the following form associates a radio with a vehicle identified by the simulation Ethernet address of its simulator:

```
Format:  attach     radio address address
Example: attach     0  address  02cf13002fd
```

An *attach* entry of the following form associates a radio with a vehicle identified by its bumper number or vehicle marking:

```
Format:  attach     radio bumper bumper-number
Example: attach     0  bumper  100
```

An *attach* entry of the following form associates a radio with an imaginary vehicle present at a specified location in the simulated world. With this form of attachment, the simulation will generate Vehicle Appearance PDUs describing the vehicle to which the radio is attached.

```
Format:  attach     radio nothing vehicle-id (x,y)
Example: attach     0  nothing  100  (40000,40000)
```

An *attach* entry of the following form associates a radio with a vehicle having a particular vehicle identifier:

```
Format:  attach     radio vid vehicle-id
Example: attach     0  vid  104
```

An *attach* entry of one of these four forms must be present for each radio implemented by the radio simulator.

*Preset*

A *preset* entry establishes single channel presets or frequency hopping data at simulator initialization time. In a situation where simulated radios are being used in a prescribed manner, preset entries may be used to eliminate the need for manually initializing single channel presets or frequency hopping data each time the simulator is restarted.

There are several forms of *preset* entry; each form is described below. The configuration file may contain any number of *preset* entries, each specifying a particular type of data for a particular radio and channel. In the descriptions that follow, the *channel* parameter identifies the channel (1 through 6) for which data is being specified.

A *preset* entry of the following form loads a channel's frequency for operation in single channel mode. The *frequency* parameter specifies a frequency in kilohertz.

```
Format:  preset      radio sc channel frequency
Example: preset      0  sc  1  40000
```

A *preset* entry of the following form loads a channel's hopset for operation in FH mode. The *hopset* parameter specifies a hopset by its integer identifier.

```
Format:  preset      radio fh channel hopset
Example: preset      0  fh  1  1234
```

A *preset* entry of the following form loads a radio's lockout set for operation in FH mode.

```
Format:  preset      radio lockout lockout-number lockout
Example: preset      0  lockout  1  1234
```

A *preset* entry of the following form loads a channel's COMSEC variable for operation in FH mode. The *comsec* parameter specifies a COMSEC value as an integer in the range 1 through 999.

```
Format:  preset      radio comsec channel comsec
Example: preset      0  comsec  1  567
```

A *preset* entry of the following form loads a radio's TRANSEC variable for operation in FH mode. The transec parameter specifies a TRANSEC value as an integer in the range 1 through 999.

```
Format:  preset      radio transec transec
Example: preset      0  transec  890
```

A *preset* entry of the following form loads a channel's time–of–day clock for operation in FH mode. The *TOD–offset* parameter specifies an offset between the channel's TOD clock and real time.

```
Format:  preset      radio tod channel TOD-offset
Example: preset      0  tod  1  0
```

*Ivis*

An *ivis* entry specifies the simulation address of the host simulating a particular IVIS unit. The address is specified as a pair of site and host numbers:

```
Format:  ivis        ivis site / host
Example: ivis        0 3/34
```

The configuration file contains one such entry for each IVIS unit associated with a simulated RIU.

*Riu*

An *riu* entry describes an RIU to be simulated. It identifies one or more radios to which the RIU is connected, and one or more IVIS units to which it is connected.

```
Format:  riu         riu number-radios { radio } number-ivis { ivis }
Example: riu         0  2  0  1  1  0
```

In this example, RIU 0 is connected to radios 0 and 1, and to IVIS unit 0.

The configuration file contains one *riu* entry for each RIU implemented by the simulator.

## A.2 Command line switches

The radio simulation program is invoked by a command that may include various optional parameters, or "switches". These switches are used primarily for controlling debugging features of the program. In normal operation the program is started without these switches and it obtains all of the configuration information it needs from the parameter file described in the previous section.

The following is a summary of the switches recognized by the simulation program. Except where indicated, all combinations of switches are valid.

`-b` *busyfile*

This switch specifies *busyfile* as a file containing a sound to be output by a radio while it is occupied with a data transfer. If this switch is absent, the file `/simnet/data/sincgars/busy-signal` is used.

`-d`

This switch enables all of the "-D" debugging options listed below. It is equivalent to the switch "-D all".

`-D all`

This switch enables all of the "-D" debugging options listed below. It is equivalent to specifying the switch "-d".

`-D loss_calc`

This switch enables the printing of debugging messages by software that computes signal attenuation due to propagation.

`-D fpt`

This switch enables the printing of debugging messages by software that emulates front panel operations.

`-D network`

This switch enables the printing of debugging messages by software that processes packets received from the simulation Ethernet.

`-D preset`

This switch instructs the simulation to use the initial channel presets specified in the configuration parameter file, or, if the parameter file specifies no presets, to use channel presets that are hard–coded in the simulation software. If this switch is absent, each radio's channel presets begin uninitialized and must be loaded explicitly by a radio operator before they can be used.

`-D riu`

This switch enables the printing of debugging messages by software that simulates RIUs.

`-D riu_buf`

This switch enables the printing of debugging messages by software that manages buffers for the RIU simulation.

`-D riu_test`

This switch enables execution of software that exercises the RIU simulation by generating artificial data traffic.

`-D state`

This switch enables the printing of debugging messages by software that tracks the state of each radio.

`-n` *noisefile*

This switch specifies *noisefile* as a file containing a sound to be output by a radio while it is in SQ–OFF mode and it is not actively receiving a signal from another radio. If this switch is absent, the file `/simnet/data/sincgars/noise-signal` is used.

`-p` *parmfile*

This switch directs the simulation to obtain configuration parameters from the file *parmfile*. If this switch is absent, the file `/simnet/data/sincgars/radio-pars` is used.

`-t`

This switch disables the use of the signal propagation model so that each of the radios simulated by the program can receive from any correctly tuned transmitter, without regard to distance or terrain.

`-v`

This switch enables the printing of debugging messages by software servicing the voice digitization hardware.

# APPENDIX B: SIMULATION RUN-TIME COMMANDS

While in operation, the SINCGARS radio simulation accepts and processes commands entered at the terminal that is used to invoke the simulation program. These commands includes ones for reporting statistics collected by the simulation, and for resetting various hardware subsystems.

The simulation prompts for entry of a command with the string "RADIO>". Each command is entered as a line consisting of one or more keywords or parameters terminated by a carriage return. Various EMACS–style line editing operations may be used to modify a command while it is being entered. For example, Ctrl–A positions the cursor at the beginning of the line, Ctrl–E positions the cursor at the end of the line, and the DELETE key erases the character to the left of the cursor. Printable characters typed are simply inserted at the cursor. Any command not understand by the program is reprinted so that the operator may edit it to correct it. Typing a "?" while entering a command causes the simulation to display a list of keywords that are valid for completing that command.

These keyboard commands are recognized:

`exit`

This command gracefully terminates the simulation.

`fp`

This command displays the status of each radio front panel, including its channel presets and switch positions.

`hardware resetsimvads`

This command resets the simulator's voice digitization hardware.

`hardware simstats`

This command displays performance statistics for the simulator's voice digitization hardware.

`hardware resetfrontpanels`

This command re–establishes synchronization between the radio simulation host and its Front Panel Adapter. Synchronization between those components of the radio simulation may be lost if power to the front panels is interrupted, or if communication between the host and the front panels is interrupted.

`help`

This command displays information about commands recognized by the simulation.

`network ethernetaddress`

This command displays the simulation Ethernet address of the radio simulation host.

`network getstats`

This command displays performance statistics for the Ethernet interface, such as the number of packets transmitted and received both with and without errors.

`network zerostats`

This command clears counters used to accumulate performance statistics for the Ethernet interface.

`riu show`

This command displays information about the configuration and performance of the RIU simulation. It identifies the RIUs simulated, and the radios and IVIS units to which they are attached. It also displays, for each RIU, counts of the following events: messages sent to the IVIS unit, messages received from the IVIS unit, complete transmissions of messages, transmission of message fragments, receipt of message fragments, retransmissions, transmissions preempted, reassembly failures, messages received with errors, duplicate messages received.

`riu zerostats`

This command clears the counters reported by the "riu show" command.

`simulation tickcount`

This command displays the number of 26–milisecond "ticks" that have occurred since the radio simulation was started.

`simulation voicechannel` *channel*

This command displays information about a particular voice digitization channel identified by its number.

`timing show`

This command displays statistics summarizing the amount of time required to perform various processing steps during each 26–millisecond voice sampling interval. Displayed are the minimum, average, and maximum time values measured during the most recent 100 frames (2.6 seconds). The processing steps

measured are: reading information from the Ethernet, simulating RIU functions, simulating ECCM remote fill functions, servicing voice digitization interfaces, and simulating front panel operations.

`timing zero`

This command clears the statistics reported by the "timing show" command.

---

NOTE: UNIX is a registered trademark of AT&T Bell Laboratories. VMEbus is a trademark of Motorola Corp. Multibus is a trademark of the Intel Corp. Ethernet is a trademark of Xerox Corp.